HP Unified Functional Testing

For the Windows ® operating systems

Software Version: 12.00

User Guide

Document Release Date: March 2014

Software Release Date: March 2014

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 - 2014 Hewlett-Packard Development Company, L.P.

Trademark Notices

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

Apple and the Apple logo are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.

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: **http://h20230.www2.hp.com/selfsolve/manuals**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: **http://h20229.www2.hp.com/passport-registration.html**

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

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

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

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

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

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

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

To find more information about access levels, go to:

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

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

About the *HP Unified Functional Testing 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.

Prerequisite Background

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

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 (http://h20230.www2.hp.com/selfsolve/manuals).

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 http://support.openview.hp.com/selfsolve/manuals.
UFT PAM	The Product Availability Matrix (PAM) provides current information about technologies and integrations supported for this version of UFT.
GUI Testing Tutorial	The GUI Testing Tutorial is a self-paced printable guide, designed to lead you through the process of creating GUI tests and familiarize you with the testing environment.
API Testing Tutorial	The API Testing Tutorial is a self-paced printable guide, designed to lead you through the process of creating API tests in the Windows 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 Viewer	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.
UFT Installation Guide	The HP Unified Functional Testing Installation Guide provides a 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.
License Server Installation Guide	The Concurrent License Server Installation Guide provides the information you need to install and maintain the HP Functional Testing Concurrent License Server.
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.

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.

Reference	Description
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.

Additional Online Resources

The following additional online resources are available from the Unified Functional Testing Help menu:

Resource	Description
HP Software Support Online	Opens the HP Software Support Web site. This site enables you to browse the HP Software Self-solve knowledge base. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. Choose **Help > HP Software Support**. The URL for this Web site www.hp.com/go/hpsoftwaresupport. • Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. • To find more information about access levels, go to: http://h20230.www2.hp.com/new_access_levels.jsp • To register for an HP Passport user ID, go to: http://h20229.www2.hp.com/passport-registration.html

Resource	Description
Testing Forums	Opens the testing forums for GUI Testing, API Testing, and BPT Testing forums. where you can interact with other users of UFT and discuss topics related to GUI Testing, API Testing, and BPT. The URLs for these sites are: • 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	Opens the HP Unified Functional Testing product page, with information and related links about UFT.
Troubleshooting & Knowledge Base	Opens the Troubleshooting page on the HP Software Support Web site where you can search the HP Software Self-solve knowledge base. Choose **Help > Knowledge Base** or **Help > Troubleshooting**. The URL for the troubleshooting Web site is http://h20230.www2.hp.com/troubleshooting.jsp.
HP Software Community	Opens the HP IT Experts Community site, where you can interact with other HP software users, read articles and blogs on HP software and access downloads of other software products.
HP Manuals Site	Opens the HP Software Product Manuals Web site, where you can search for the most up-to-date documentation for a selected HP Software product. The URL for this Web site is http://support.openview.hp.com/selfsolve/manuals (requires an HP Passport).
What's New	Opens the UFT What's New Help, describing the new features and enhancements in this version of UFT.
Product Movies	Opens a page on HPLN (HP Live Networks) displaying a list of all product movies.
HP Software Web site	Opens the HP Software Web site. This site provides you with the most up-to-date information on HP Software products. This includes new software releases, seminars and trade shows, customer support, and more. The URL for this Web site is www.hp.com/go/software.

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.
- Mercury Flight application. To access from the Start menu, select **All Programs > HP Software > HP Unified Functional Testing > Sample Applications > Flight API / Flight GUI**.

Part 1: UFT Introduction

Chapter 1: UFT Introduction

Relevant for: GUI testing and API testing

This chapter includes:

HP Unified Functional Testing Overview

Relevant for: GUI testing and API testing

Welcome to HP Unified Functional Testing (UFT), the advanced solution for functional test and regression test automation, combined with functional testing for headless systems.

UFT's combined solution for GUI and API (service) testing enables you to test functionality across multiple application layers, such as the front-end GUI layer as well as back-end service layers. Additionally, the integrated BPT features enable a wider range of both technical and non-technical UFT users, maximizing your opportunity to create comprehensive automated tests.

UFT helps you to cut time-to-market by starting automated functional testing earlier in your process. You can also streamline your tools, cutting costs on both training and licenses.

UFT provides you with an integrated system for the following main services:

- "UFT GUI Testing - Overview" on the next page
- "UFT API Testing - Overview" on page 69
- "Business Process Testing in UFT - Overview" on page 72

You can also integrate GUI testing and API testing in a single test, calling tests of one type from the other, as described in "UFT Integrated Testing - Overview" on page 70.

> **Note:** UFT supports creating and managing tests and components, for both GUI testing and API testing. Some sections of this help are relevant only for certain types of UFT functionality, such as tests, GUI tests and components, or actions. Each section of the help is marked with the relevant area of functionality.

Accessibility

Many operations are performed using the mouse. In accordance with Section 508 of the W3C accessibility standards, UFT also recognizes operations performed using the **MouseKeys** option in the Windows Accessibility Options utility. Additionally, you can perform many operations using shortcut keys.

Unicode Compliancy

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

UFT GUI Testing - Overview

Relevant for: GUI tests and components

UFT's GUI testing solution deploys the concept of keyword-driven testing to enhance test creation and maintenance.

Keyword-driven testing is a technique that separates much of the programming work from the actual test steps. This enables you to create test steps earlier, and maintain them with only minor updates, even when there are significant changes in your application or your testing needs.

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

- Advanced testers who are familiar with VBScript can also use the Editor, which enables you to add and update statements that enhance your test with programming. Use the Editor to increase the test's power and flexibility.

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

For more details about supported functionality for GUI tests, see "Additional GUI Testing Capabilities and Tools" on page 67.

Basic GUI Test Structure

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

- Each action is comprised of steps. As you add steps to an action, they are displayed in the UFT window, either as part of the Keyword View or the Editor. In the Keyword View, every step includes automatically generated documentation that provides a plain language textual description of what the step does.

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

Creating and Enhancing Your GUI Test

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

- You can also create function libraries and call their functions from your test. For example, you can define functions and use them as keywords in your test.

For details about the steps in the GUI testing process, see "GUI Testing Workflow" below.

GUI Testing Workflow

Relevant for: GUI tests and components

This section describes the general workflow of the GUI testing process, which involves the following main stages:

- "Stage 1: Analyzing Your Application" below
- "Stage 2: Preparing the Testing Infrastructure" below
- "Stage 3: Adding Steps to Your Actions" on the next page
- "Stage 4: Enhancing Your Test" on page 65
- "Stage 5: Running and Debugging Your Test" on page 65
- "Stage 6: Analyzing Run Results and Reporting Defects" on page 66

Stage 1: Analyzing Your Application

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

- **Determine the development environments in which your application controls were developed.** Development environments might include Web, Java, or .NET, and so on. You must load the required UFT add-ins when you open UFT, and before creating your test.

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

- **Decide how to divide these processes into smaller units that will be represented by your test's actions.** Each action should emulate an activity that a customer might perform when using your application.

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

Stage 2: Preparing the Testing Infrastructure

To complete the infrastructure that is part of the planning process, you need to do the following:

- **Build the set of resources to be used by your tests.** Resources can include shared object repositories containing test objects (which are representations of the objects in your application), function libraries containing functions that enhance testing functionality, and so on.

For details, see "Managing Test Objects in Object Repositories" on page 1198 and "User-Defined Functions and Function Libraries" on page 1029.

- **Configure UFT according to your testing needs.** This can include setting up your global testing preferences, your run session preferences, any test-specific preferences, and recovery scenarios. You can also create automation scripts that automatically set the required configurations (such as the add-ins to load) on the UFT client at the beginning of a run session.

  For details, see "UFT Automation Scripts" on page 1155.

- **Create one or more tests that serve as action repository tests.** You can create an action repository test to store reusable actions to be used in your tests. Generally, you create an action repository test for each area of your application to be tested. Storing all of your actions in specific tests enables you to maintain your actions in a central location.

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

  When you create your tests, you insert calls to one or more of the actions stored in this repository. When you update an action in the action repository, the update is reflected in all tests that contain a call to that action. When you run a test, only the relevant action repository tests are loaded.

Stage 3: Adding Steps to Your Actions

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

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

You can create steps using the keyword-driven functionality available in the table-like, graphical Keyword View—or you can use the Editor, if you prefer to program steps directly in VBScript. You can also add steps to your test using the following additional methods:

- **Drag objects.** Drag objects from your object repository or from the Toolbox pane to add keyword-driven steps in the Keyword View or Editor. The object repository and Toolbox pane contain all of the objects that you want to test in your application.

  When you drag an object into the Keyword View, a step is created in the action with the default operation for that object. For example, if you drag a **button** object into the Keyword View, the **Click** operation is automatically defined for the step. You can then modify the step as needed. For more details, see "Keyword View" on page 919 and "Toolbox Pane User Interface (GUI Testing)" on page 510.

  Advanced users can also add steps using the Editor, as described in "Programming in GUI Testing Documents in the Editor" on page 994.

- **Record on your application.** As you navigate through your application during a recording session, each step you perform is displayed as a row in the Keyword View. A step is something that causes or makes a change in your application, such as clicking a link or image, or submitting a data form. In the Editor, these steps are displayed as lines in a test script (VBScript). The Documentation column of the Keyword View displays a description of each step in easy-to-understand sentences. For more details, see "Keyword View" on page 919.

Stage 4: Enhancing Your Test

You can enhance the testing process by modifying your test with special testing options and/or with programming statements. For example:

- **Insert checkpoints and output values into your test.** A **checkpoint** checks specific properties or other characteristics of an object and enables you to identify whether or not your application is functioning correctly. For details, see "Checkpoints Overview" on page 1396.

- **Use output values to extract data from your test.** An **output value** is a value retrieved during the run session and entered into your data table or stored in a variable or a parameter. You can subsequently use this output value as input data in your test. This enables you to use data retrieved during a run session in other parts of the test. For details, see "Output Values Overview" on page 1488.

- **Replace fixed values with parameters to broaden the scope of your test.** When you test your application, you can **parameterize** your steps to check how your application performs the same operations with different data. You may supply data in the data table, define environment variables and values, define test or action parameters and values, or insert a step that generates random numbers for current user and test data. For details, see "Parameterizing Object Values" on page 1526.

- **Add user-defined functions by creating function libraries and calling their functions from your test.** Function libraries contain VBScript functions, subroutines, statements, and so on. You can call functions from your test, or use functions as operations in your test or component. For details, see "User-Defined Functions and Function Libraries" on page 1029.

- **For complex testing goals:** Use other UFT features to enhance your test, and/or programming statements. For details, see "Generated Programming Operations" on page 968.

Stage 5: Running and Debugging Your Test

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

- **Debug the test.** You can control your run session to help you identify and eliminate defects in your test, as follows:

  - Use the **Step Into**, **Step Over**, and **Step Out** commands to run your test step by step.

  - Begin your run session from a specific step in your test, or run the test until a specific step is reached.

  - Set breakpoints to pause your test at predetermined points. You can view or change the value of variables in your test each time it stops at a breakpoint in the Debug Viewer.

  - Manually run VBScript commands in the Debug Viewer.

  For more details, see "Debugging Tests and Components" on page 674.

- **Test your application.** The test starts running from the first line in your test and stops at the end of the test.

  While running, UFT connects to your application and performs each operation in your test, including any checkpoints, such as checking text strings, objects, tables, and so forth.

  If you parameterized your test with data table parameters, the test (or specific actions in your test) repeats for each set of data values in the data table.

  For more details, see "Running Tests and Components" on page 634.

- **Update your test.** If you know that your application has changed, run your test in one of the following modes to update your test accordingly:

  - **Maintenance Run Mode.** Use this mode if you expect that UFT will not be able to identify the objects in your test. As your test runs, UFT opens a wizard for each object that cannot be found in the application, to guide you through resolving each issue. When the object can be identified and the issue is resolved, the run continues.

  - **Update Run Mode.** Use this mode to update the any of the following:

    - Property sets used for test object descriptions

    - Expected checkpoint values

    - Data available to retrieve in output values

    - Active Screen images and values

  For more details, see "Maintaining and Updating GUI Tests or Components " on page 1078.

Stage 6: Analyzing Run Results and Reporting Defects

After you run your test, you can view the results of the run in the Run Results Viewer, including both a summary and a detailed report, and you can report defects detected during the run session.

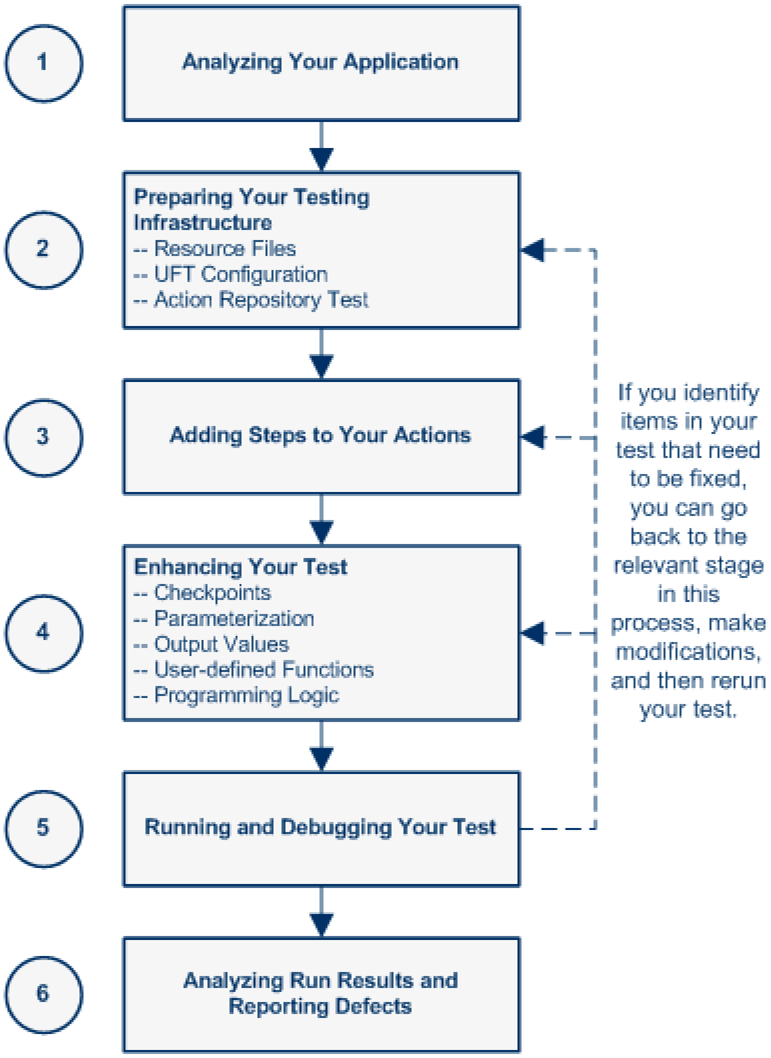
- **View still images or movies.** If you captured still images or movies of your application during the run, you can view these from the Screen Recorder tab of the Run Results Viewer.

- **Local system monitoring.** If you enabled local system monitoring for your test, you can view the results in the System Monitor tab of the Run Results Viewer.

- **Report defects in ALM.** If you have access to ALM, the HP centralized quality solution, you can report the defects you discover to the project database.

- **Report defects automatically or manually.** You can add steps to your test that automatically report each failed step in your test, or you can report them manually from the Run Results Viewer.

For details about run results, see the *HP Run Results Viewer User Guide*. For details about defect reporting, see "ALM Integration" on page 707.


1
Analyzing Your Application
2
Preparing Your Testing Infrastructure
-- Resource Files
-- UFT Configuration
-- Action Repository Test
3
Adding Steps to Your Actions
4
Enhancing Your Test
-- Checkpoints
-- Parameterization
-- Output Values
-- User-defined Functions
-- Programming Logic
5
Running and Debugging Your Test
6
Analyzing Run Results and Reporting Defects
If you identify items in your test that need to be fixed, you can go back to the relevant stage in this process, make modifications, and then rerun your test.

Additional GUI Testing Capabilities and Tools

Relevant for: GUI tests and components

The following sections describe additional UFT functionality for GUI tests:

- "Testing Objects from Different Environments" on the next page
- "Programming in the Editor " on the next page

- "Functions and Function Libraries" below

Testing Objects from Different Environments

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

- Loading the relevant add-in enables UFT to recognize and learn the objects in your application so that you can design automated tests that mirror your customer's operations and business processes. You can then run these tests to check that your application works as expected.
- You load add-ins using the Add-in Manager dialog box described in "How to Start UFT" on page 114. For details on the on the Add-in Manager dialog box and each of the UFT add-in environments, see the *HP Unified Functional Testing Add-ins Guide*.

Programming in the Editor

The Editor displays a text-based version of your action.

The test is composed of statements written in VBScript (Microsoft Visual Basic Scripting Edition) that correspond to the steps and checks displayed in the Keyword View. You can move between the two views by selecting the Editor / Keyword View toggle button. For details, see "Programming in GUI Testing Documents in the Editor" on page 994.

For more details on the test objects and methods available for use in your test and how to program using VBScript, see the *HP UFT Object Model Reference for GUI Testing* and the *VBScript Reference* (select **Help > HP Unified Functional Testing Help**).

Functions and Function Libraries

If you have sets of steps that are repeated in several actions or tests, you may want to consider creating and using user-defined functions.

- A user-defined function encapsulates an activity (or a group of steps that require programming) into a keyword, also called an operation. User-defined functions help shorten your tests, and make them easier to design, read, and maintain.
- You can use the built-in Editor view to create and edit user-defined functions during your session. A function library is a Visual Basic script containing VBscript functions, subroutines, modules, and so forth. You can also use the Function Definition Generator to assist you in defining new functions.
- When you create a function, you can insert it directly in an action to make it available only within that action, or you can insert it in a function library to make it available to any test that is associated with that function library. For details, see "User-Defined Functions and Function Libraries" on page 1029.

UFT API Testing - Overview

Relevant for: API tests

UFT's API (service) testing solution provides tools for the construction and execution of functional tests for headless (GUI-less) systems. For example, you can use UFT to test standard Web Services, non-SOAP Web Services, such as REST, and so on.

You create an API test by dragging and dropping activities from the UFT Toolbox pane into the test, displayed in the canvas. The toolbox provides a collection of activities for functional testing in areas such as REST, Web Services, JMS, and HTTP. You can add more activities to the toolbox by importing WSDLs or providing other contract definitions.

For more details about service (API) testing in UFT, including all of the optional steps included in the testing workflow, see "API Test Creation Overview" on page 1589.

UFT Integrated Testing - Overview

Relevant for: GUI testing and API testing

Integration for GUI and API Testing

You can integrate your GUI and service testing processes in a single test by including calls from your GUI test to API tests, or from your API tests to GUI tests. When you insert a call to another test, the call is displayed as nested under the relevant action in the canvas.

- You insert and modify calls to API tests from GUI tests using the "Call to API Test/Action Dialog Box" described on page 1075. For details, see "Integration with API Tests" on page 1067.
- You insert and modify calls to GUI tests from API tests by dragging the Call GUI Action or Test onto the canvas from the HP Automated Testing Tools node in the Toolbox pane. For more details, see "How to Call External Tests or Actions" on page 1639.

Integration with ALM

You can also use UFT together with ALMto manage the entire testing process. For example, you can use ALM to:

- Create a project (central repository) of manual and automated tests
- Build test cycles
- Run tests
- Report and track defects

You can also create reports and graphs to help you review the progress of test planning, runs, and defect tracking before a software release.

Tests and components created in UFT can be saved directly to your ALM project, and you can run UFT tests and review and manage the results in ALM. For details see "ALM Integration" on page 707, and the *HP Application Lifecycle Management User Guide*.

> **Note:** Unless otherwise specified, references to **Application Lifecycle Management** or **ALM** in this guide apply to all currently supported versions of ALM and Quality Center. Note that some features and options may not be supported in the specific edition of ALM or Quality Center that you are using.
>
> For a list of the supported versions of ALM or Quality Center, see the *HP Unified Functional Testing Product Availability Matrix*, available from the UFT Help or the root folder of the Unified Functional Testing DVD. The most up-to-date product availability matrix is available from the HP Software Product Manuals site, at http://h20230.www2.hp.com/selfsolve/manuals (requires an HP Passport).

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

Business Process Testing in UFT - Overview

Relevant for: business process tests and flows

Business Process Testing (BPT) works within UFT or ALM as a component-based testing framework. Working with a testing framework provides many advantages to enterprises, including streamlining the creation and maintenance of both manual and automated tests, and maximizing efficiency for testing complete business processes.

Business components:

- Enable structured automated testing.
- Reduce the duplication of effort when combining manual tests with automatic tests.
- Enable component reusability to speed-up the automation process.
- Provide the ability to pass parameters from one step to another within your business process. You can save the output of a step to a parameter and use it as an input value for subsequent steps.
- Simplify on-going test maintenance.
- Minimizes time-to-test.

"Business Process Testing in UFT" on page 2062 describes how to use BPT in UFT, including how to maintain business process tests and flows, business components, and application areas in UFT.

UFT Program Management

The following sections describe how you can manage your UFT program, including available licenses and access levels.

To check for software updates, patches, or service packs for UFT, visit the HP Software Support Site. The **HP Update** program does not provide updates for UFT.

Licensing

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

- A permanent **seat** license that is specific to the computer on which it is installed (includes a 30-day demo license)
- A network-based **concurrent** license that can be used by multiple UFT users

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

For information on modifying your license information, see the *HP Unified Functional Testing Installation Guide*.

> **Note:** You can also open UFT using a legacy license, although the functionality will be limited to the service that you are licensed to use. For example, you can open UFT using a legacy QuickTest Professional or Service Test license and access GUI testing or API testing functionality.

Access Permissions

You must make sure the following access permissions are set to run UFT or to work with ALM.

Permissions Required When Working with UFT

You must have the following file system permissions:

- Full read and write permissions to the Temp folder
- Read permissions to the Windows folder and to the System folder
- Full read and write permissions to the folder on which you are saving solutions, tests, or run results
- Full read and write permissions to the <Program Files>\Common Files\Mercury Interactive folder

- If you are working on the Windows Vista, Windows 7, or Windows Server 2008 operating systems: Full read and write permissions to the <Program Data>\HP folder
- Full read and write permissions to the User Profile folders
- Full read and write permissions to the <Windows>\mercury.ini file
- Full read and write permissions to the following AppData folders:
    - %userprofile%\AppData\Local\HP
    - %appdata%\Hewlett-Packard\UFT
    - %appdata%\HP\API Testing

    > **Note:** Read/write permissions to these folders should also enable permission to any subfolders contained in the folders listed above. If not, the system administrator must grant administrative permissions to the subfolders contained in these folders.

You must have the following registry key permissions:

- Full read and write permissions to the keys uder HKEY_CURRENT_USER\Software\Mercury Interactive or [HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Hewlett-Packard]
- Full read and write permissions to all the keys under HKEY_CURRENT_USER\SOFTWARE\Hewlett Packard
- Read and Query Value permissions to all the HKEY_LOCAL_MACHINE and HKEY_CLASSES_ROOT keys

Permissions Required When Working with ALM

You must have the following permissions to use UFT with ALM:

- Full read and write permissions to the ALM cache folder
- Full read and write permissions to the <Program Data>\HP folder
- Full read and write permissions to the UFT Add-in for ALM installation folder
- Administrative permissions for the first connection to ALM

> **Note:** If you do not have administrative permissions, you can turn off the UAC for your user account instead in order to perform the first connection to ALM.

Permissions Required When Working with BPT

The ALM Project Administrator can control access to a project by defining which users can log in to it and by specifying the types of tasks each user may perform. The ALM Project Administrator can assign permissions for adding, modifying, and deleting folders, components, steps, and parameters in the Business Components module of an ALM project.

You need to make sure you have the required ALM permissions before working with business components and application areas.

- To work with component steps in ALM, you must have the appropriate **Add Step**, **Modify Step**, or **Delete Step** permissions set. You do not need **Modify Component** permission to work with component steps. The **Modify Component** permission enables you to work with component properties (the fields in the component Details tab).
- To work with parameters in ALM or in a testing tool, you must have all the parameter task permissions set in ALM.
- To modify application areas, you must have the separate permissions for resources required for modifying components, and adding, modifying, and deleting steps. All four permissions are required. If one of these permissions is not assigned, you can open application areas only in read-only format.

For more information on setting user group permissions in the Business Components module, see the *HP Business Process Testing User Guide*.

Demo Application(s)

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

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

A sample Flight Windows-based application is also provided with the UFT installation. You can access it from **Start > Programs > HP Software > HP Unified Functional Testing > Sample Applications > Flight API** or **Flight GUI**.

> **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" below.

Accessing UFT in Windows 8 Operating Systems

UFT applications and files that were accessible from the **Start** menu in previous versions of Windows are accessible in Windows 8 from the **Start** screen or the **Apps** screen.

- **Applications (.exe files).** You can access UFT applications in Windows 8 directly from the **Start** screen. For example, to start UFT, double-click the **HP Unified Functional Testing** shortcut .

Other examples of applications accessible from the **Start** screen include:

- 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 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 in 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.

Chapter 2: UFT at a Glance

Relevant for: GUI tests and components and API testing

This chapter includes:

Concepts

UFT at a Glance - Introduction

Relevant for: GUI tests and components and API testing

UFT provides a unified solution to test your application, combining the ability to test the user interface of your application (through GUI testing) and the non-GUI sections of your application (through API testing).

You can create a test of your application from start to finish, or you can test sections of it through the creation of individual actions or components to test each process in your application or application sections. You can then combine the individual process testing scenarios into a test (for actions) or a business process test or flow (for components).

UFT also works with ALM, HP's centralized quality control solution. You can save tests, components, function libraries and application areas with your ALM project and make them available to multiple users and testing projects.

UFT Main Window Overview

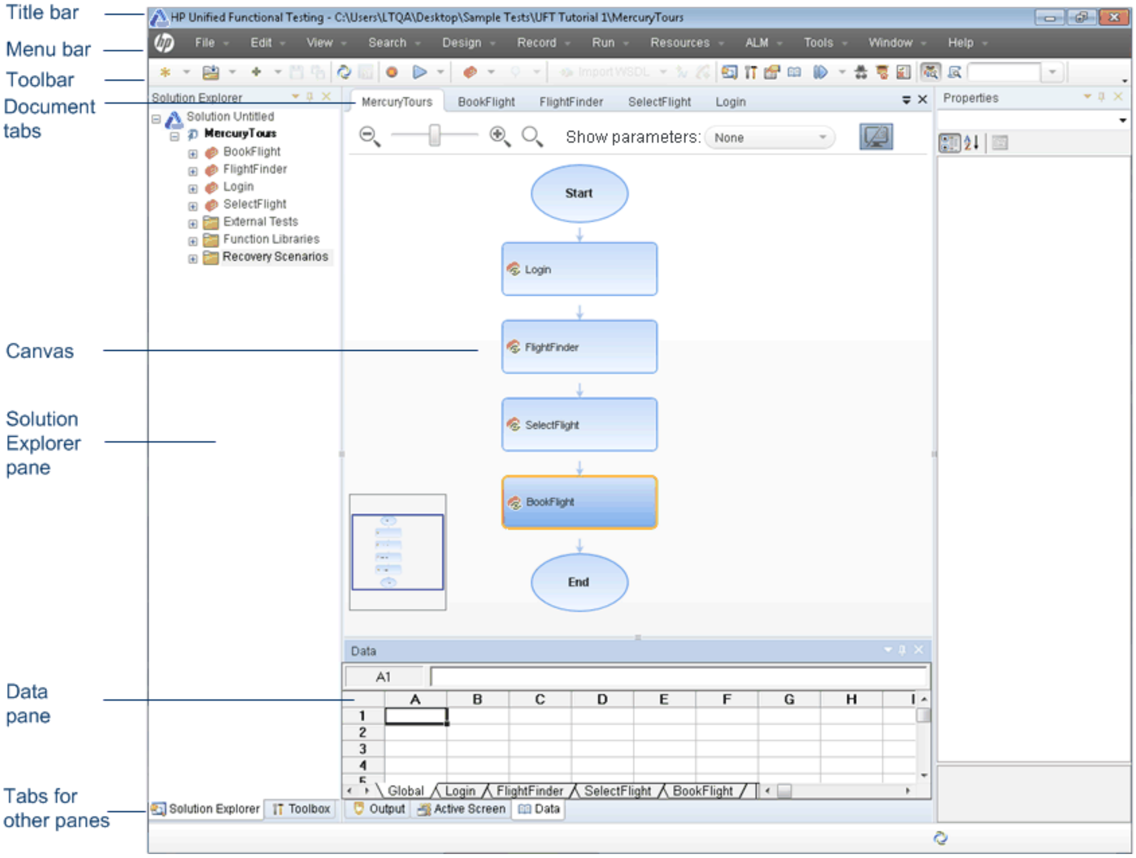
Relevant for: GUI testing, API testing, and Business Process Testing

The main window in UFT provides multiple areas to design, edit, debug, and run testing documents. The following table introduces the key elements in the UFT window:

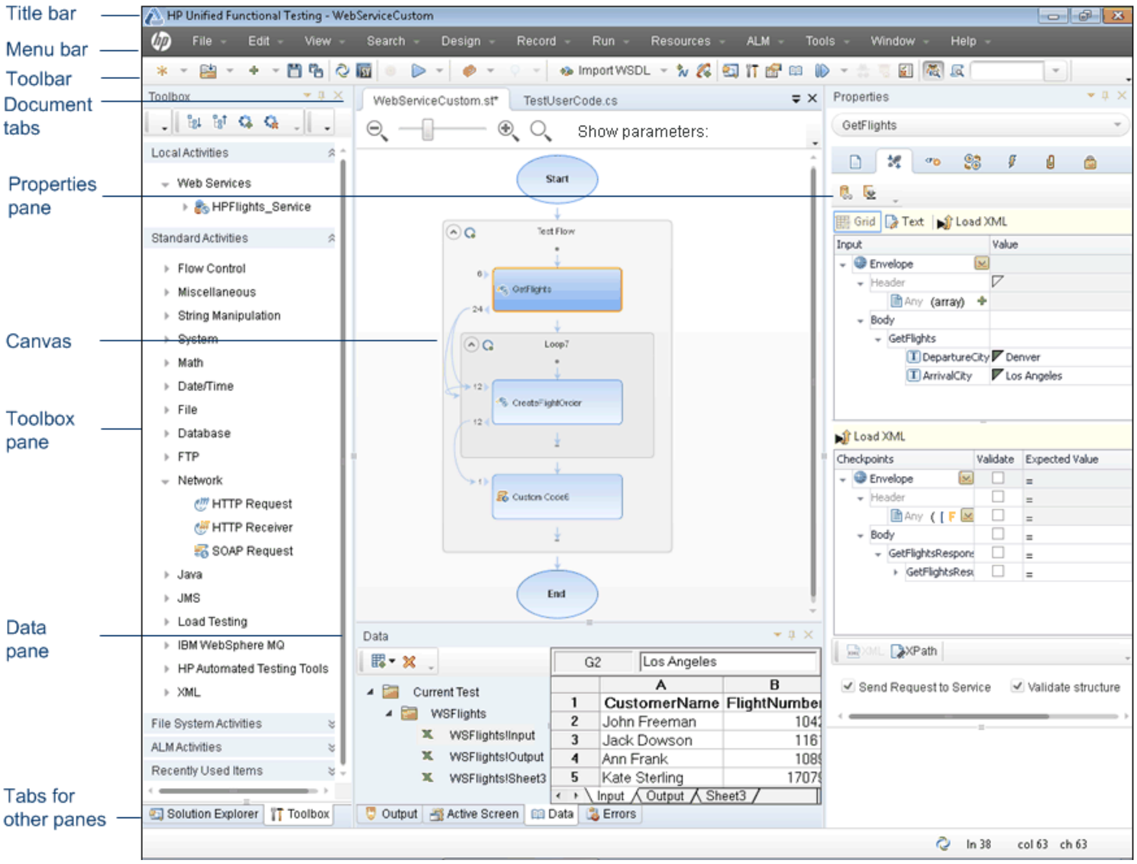
Element	Description
"Start Page"	Welcomes you to UFT. You can use the shortcut buttons to open new and existing documents. For details, see "Start Page" on page 121.
Document Pane	The main area to design and edit testing documents. • **Canvas.** Visually displays the test flow as a series of actions (for GUI tests) or steps (for API tests). For details, see "The Canvas" on page 209. • **Editor.** Displays each test step as a line of code and enables you to create GUI actions or function libraries with VBScript code or API user code using C# code. For details, see "Editor at a Glance" on page 82. • **Keyword View (GUI testing only).** Displays each step and the object hierarchy in a modular, icon-based table and enables you to create test steps using a keyword methodology. For details, see "Keyword View Overview" on page 921. For details on the different types of documents used in UFT, see "Editor at a Glance" on page 82.

Element	Description
"UFT Panes at a Glance"	Provide information and functionality about the current test, business component, function library or application area. For details, see "UFT Panes at a Glance" on page 94.

The image below shows the UFT main window when a GUI test is open.



The image below shows the UFT main window when an API test is open.



Input	Value
Envelope	
Header	
Any (array)	
Body	
GetFlights	
DepartureCity	Denver
ArrivalCity	Los Angeles

G2	Los Angeles

	A	B
1	CustomerName	FlightNumbe
2	John Freeman	104
3	Jack Dowson	116
4	Ann Frank	108
5	Kate Sterling	1707

For details about the key areas displayed in the main window, as well as details about other types of UFT documents, see the Glance subsections in the remainder of this chapter.

Canvas at a Glance

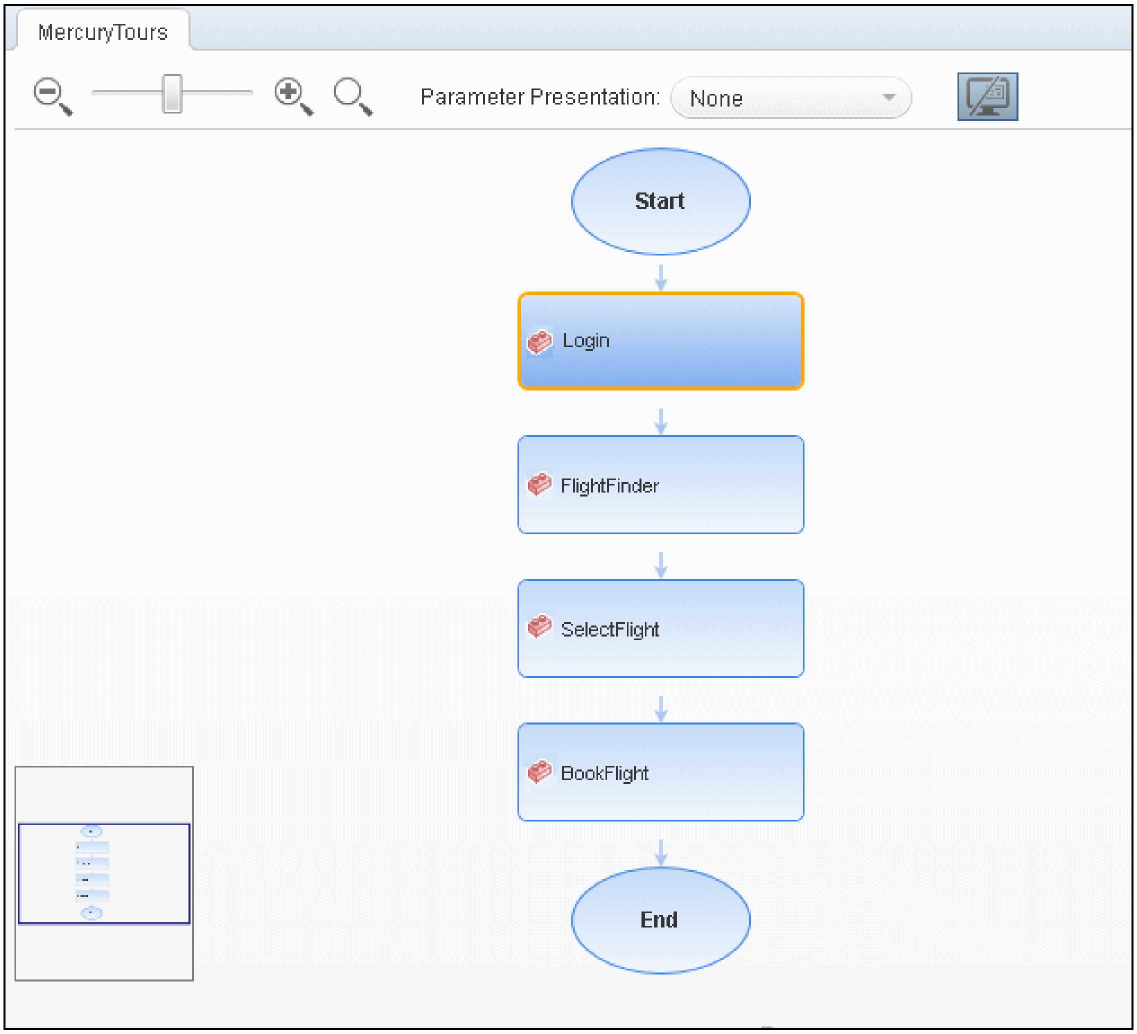
Relevant for: GUI tests only and API testing

The canvas provides a visual representation of the test flow.

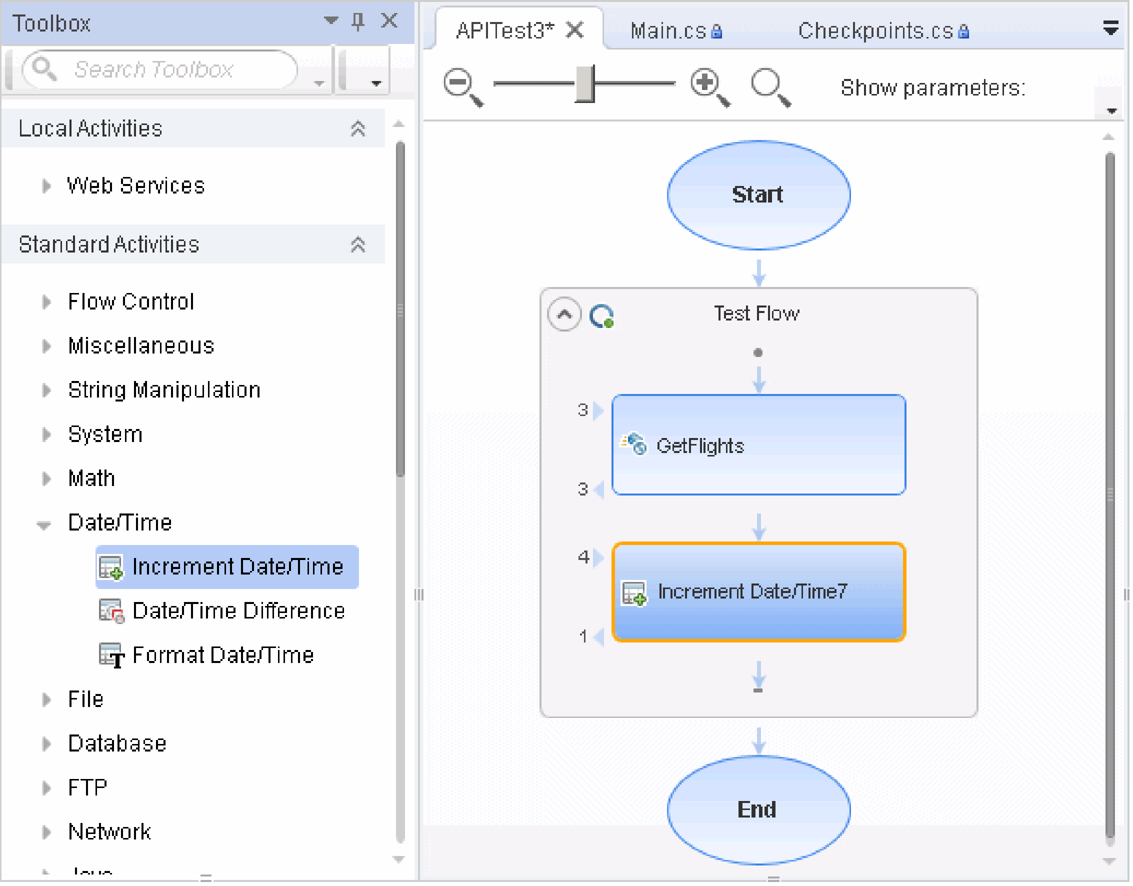
For GUI testing: The canvas displays a flow of the action calls found within a test, including calls to copies of actions or calls to existing actions. You can double-click each action to open the action as a separate, editable document tab.

For API testing: The canvas displays a flow of the steps found within a test, including parameter links between steps. You add steps to the canvas by dragging them from the **Toolbox** pane.

The image below shows the canvas for a **GUI test**.



The image below shows the canvas for an **API test**.



For details on the canvas, see "The Canvas" on page 209.

Editor at a Glance

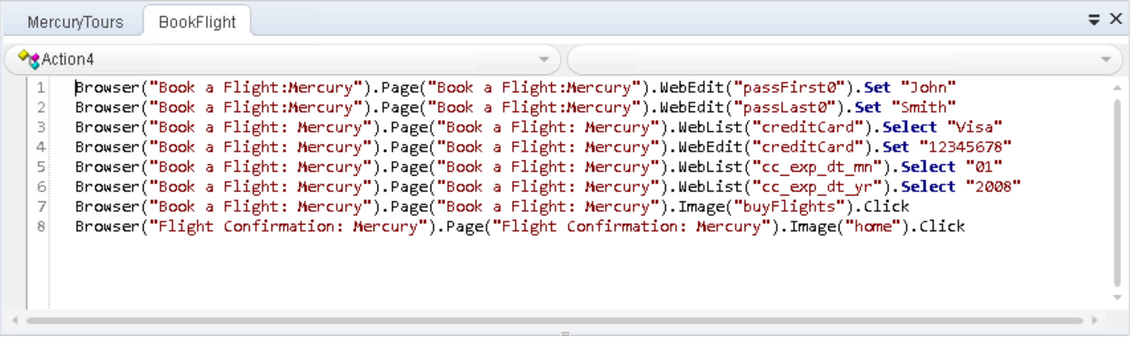
Relevant for: GUI tests, scripted GUI components, function libraries, and user code files

The Editor displays test steps as lines of code and enables you to add steps to a document or user code file using VBScript code (for GUI tests and components) or C# code (for API tests and components).

For GUI testing: In the Editor, UFT displays each operation performed on your application in the form of a script, comprised of VBScript statements. For each object and method in an Editor statement, a corresponding row exists in the Keyword View.

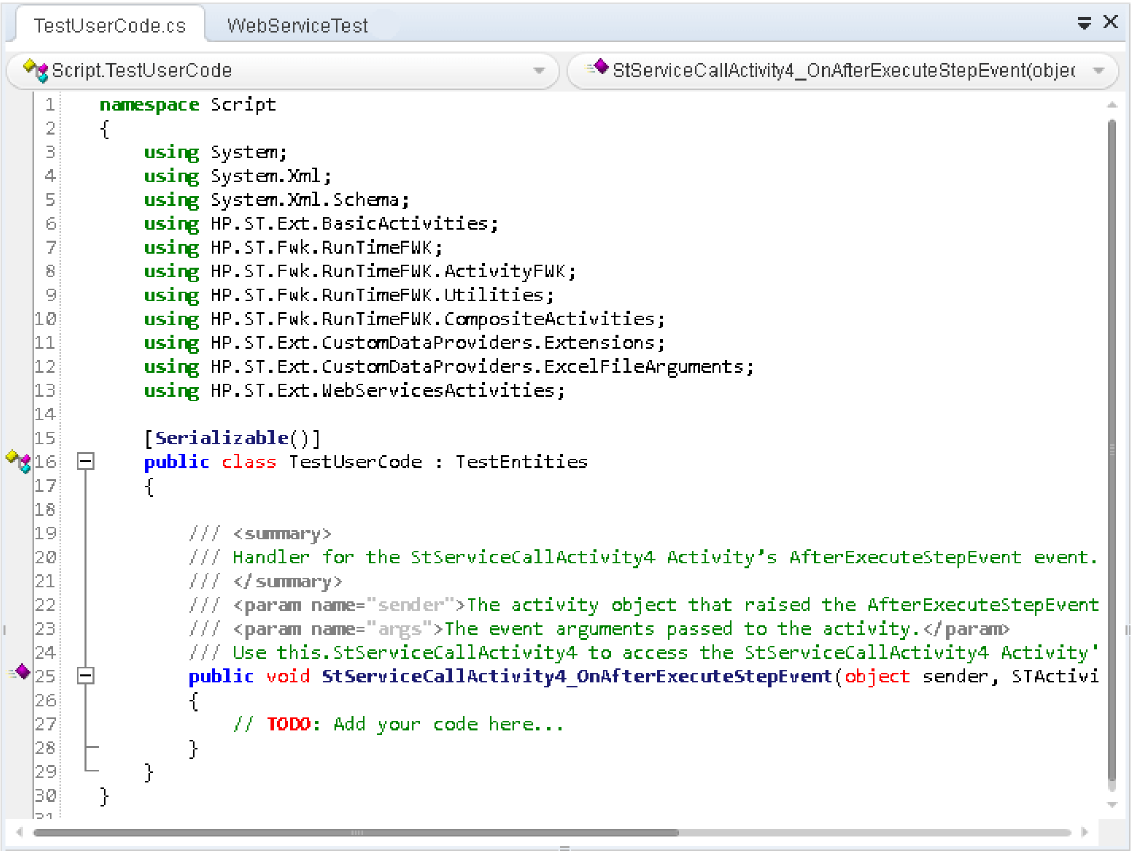
For API testing: The Editor enables you to write customized user code forAPI tests run during test steps or to write custom code which defines the properties and behavior of different test steps.

The image below shows the Editor for **GUI testing**.



```
Browser("Book a Flight:Mercury").Page("Book a Flight:Mercury").WebEdit("passFirst0").Set "John"
Browser("Book a Flight:Mercury").Page("Book a Flight:Mercury").WebEdit("passLast0").Set "Smith"
Browser("Book a Flight: Mercury").Page("Book a Flight: Mercury").WebList("creditCard").Select "Visa"
Browser("Book a Flight: Mercury").Page("Book a Flight: Mercury").WebEdit("creditCard").Set "12345678"
Browser("Book a Flight: Mercury").Page("Book a Flight: Mercury").WebList("cc_exp_dt_mn").Select "01"
Browser("Book a Flight: Mercury").Page("Book a Flight: Mercury").WebList("cc_exp_dt_yr").Select "2008"
Browser("Book a Flight: Mercury").Page("Book a Flight: Mercury").Image("buyFlights").Click
Browser("Flight Confirmation: Mercury").Page("Flight Confirmation: Mercury").Image("home").Click
```

The image below shows the Editor (open to an event handler) for **API testing**.



```
namespace Script
{
    using System;
    using System.Xml;
    using System.Xml.Schema;
    using HP.ST.Ext.BasicActivities;
    using HP.ST.Fwk.RunTimeFWK;
    using HP.ST.Fwk.RunTimeFWK.ActivityFWK;
    using HP.ST.Fwk.RunTimeFWK.Utilities;
    using HP.ST.Fwk.RunTimeFWK.CompositeActivities;
    using HP.ST.Ext.CustomDataProviders.Extensions;
    using HP.ST.Ext.CustomDataProviders.ExcelFileArguments;
    using HP.ST.Ext.WebServicesActivities;

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the StServiceCallActivity4 Activity’s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEvent
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.StServiceCallActivity4 to access the StServiceCallActivity4 Activity'
        public void StServiceCallActivity4_OnAfterExecuteStepEvent(object sender, STActivi
        {
            // TODO: Add your code here...
        }
    }
}
```

For details on using the Editor, see "Editing Text and Code Documents" on page 305.

Keyword View at a Glance

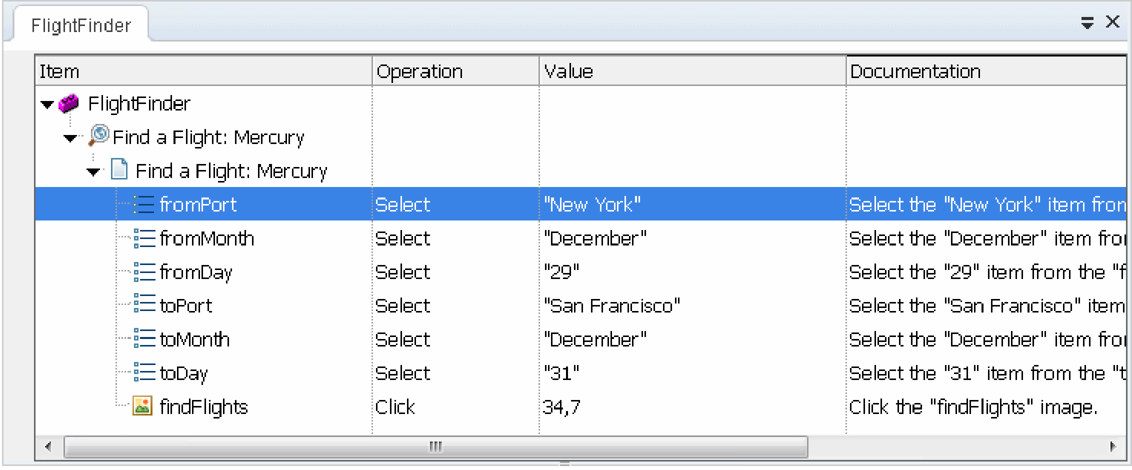
Relevant for: GUI tests and keyword GUI components

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

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

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

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



Item	Operation	Value	Documentation
FlightFinder			
Find a Flight: Mercury			
Find a Flight: Mercury			
fromPort	Select	"New York"	Select the "New York" item fron
fromMonth	Select	"December"	Select the "December" item fro
fromDay	Select	"29"	Select the "29" item from the "f
toPort	Select	"San Francisco"	Select the "San Francisco" item
toMonth	Select	"December"	Select the "December" item fro
toDay	Select	"31"	Select the "31" item from the "t
findFlights	Click	34,7	Click the "findFlights" image.

For details on using the Keyword View, see "Keyword View" on page 919.

UFT Testing Documents

Relevant for: GUI testing and API testing

In UFT, you can create, edit, debug, and run a variety of testing documents. The following document types are available in UFT:

Document	Description
GUI Test (**GUI testing only)**	Tests your application's user interface using actions, steps, and resources. For details, see "GUI Test Creation Overview" on page 812.
API Test	Tests your application's non-GUI, service-based functions. For details, see "UFT API Testing Overview" on page 1590.

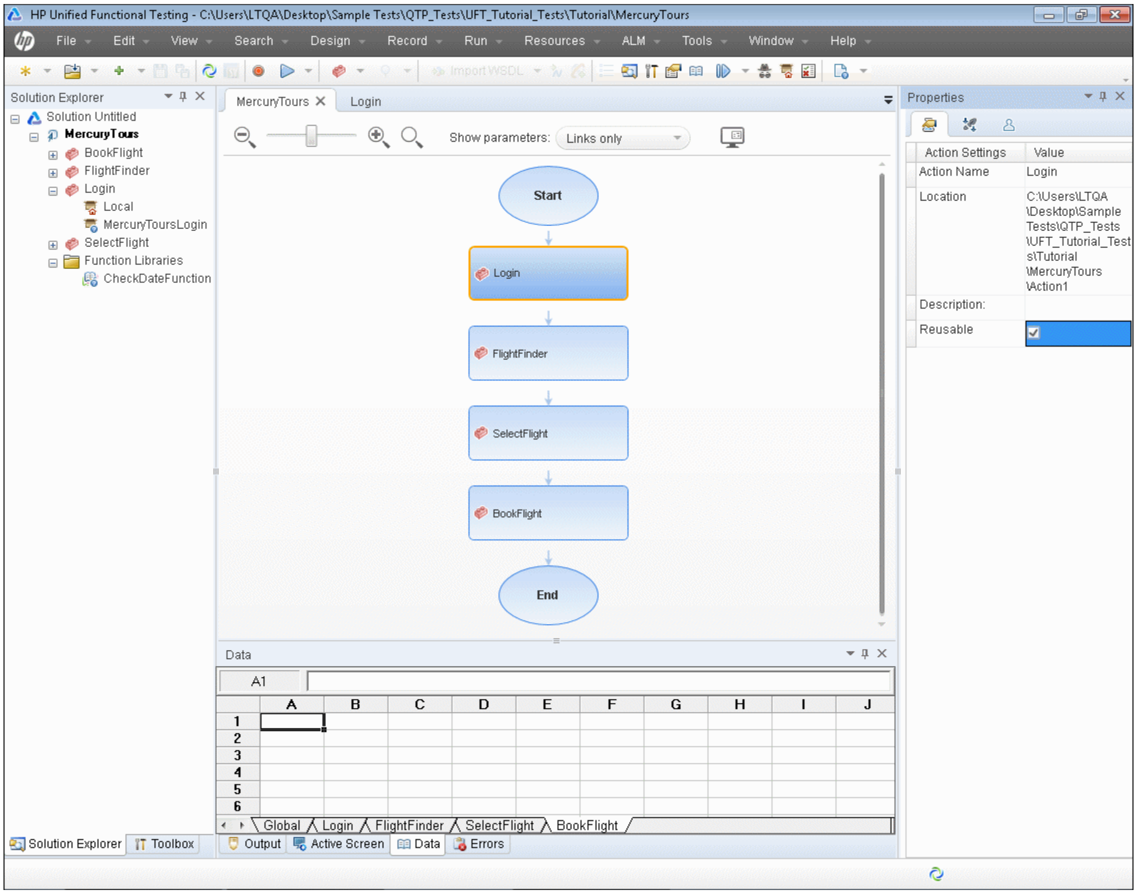
Document	Description
Keyword GUI Component (**GUI testing only)**	Tests sections of your application as a business component using keywords and operations. For details, see "Keyword GUI Components Overview" on page 2084.
Scripted GUI Component (**GUI testing only)**	Tests a section of your application as business component using script commands in the Editor or keywords in the Keyword View. For details, see "Scripted GUI Components Overview" on page 2085.
API Component	Creates a business component similar to an API test. Certain limitations apply to API components. For details about API testing, see"UFT API Testing Overview" on page 1590. For details about exceptions when working with API components, see "Working with API components" on page 2079.
Business Process Tests	Tests your application by combining multiple components into a test with an ALM project. For details, see "Business Process Testing in UFT - Overview" on page 2065.
Business Process Flows	Creates a test flow of multiple components to test a specific order of steps in your application with an ALM project. For details, see "Business Process Testing in UFT - Overview" on page 2065.
Application Area (**GUI testing only)**	Creates a resource area for your application by defining resources and settings for the component testing a specific section of your application. For details, see "Application Areas - Overview" on page 2096.
Function Library (**GUI testing only)**	Creates a document containing functions and subroutines to use with your GUI test or component. For details, see "Function Library Overview" on page 1030.
User Code Files (**API testing only)**	Creates a document containing functions or classes to use with yourAPI test or component. For details, see "Writing Code for API Test Events - Overview" on page 1938.

GUI Tests at a Glance

Relevant for: GUI tests only

GUI tests enable you to test user interface elements in your application. You can design tests with a keyword-driven approach through the Keyword View or by writing VBScript test scripts in the Editor. Each test is comprised of single or multiple calls to actions, which each contain individual test steps. After building your test, UFT runs each step individually and reports which steps were performed, as well as which steps succeeded or failed.

The image below shows a GUI test displaying the canvas tab. Each action is also displayed as its own editable tab.



Action Settings	Value
Action Name	Login
Location	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Tests\Tutorial\MercuryTours\Action1
Description:	
Reusable	☑

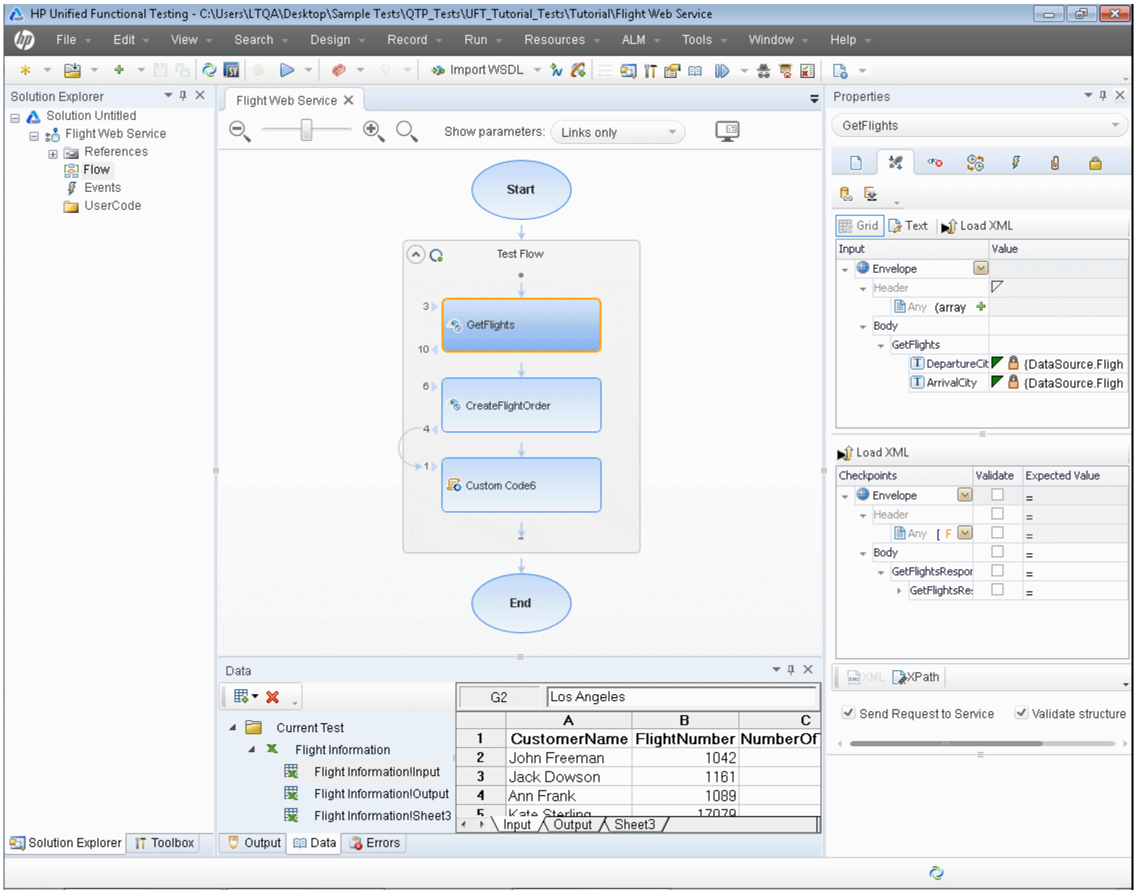
For details, see "Methodologies for Creating Tests" on page 813.

API Tests at a Glance

Relevant for: API tests

API tests enable you to test your GUI-less applications (also called headless applications) or the sections of your application which lack user interface elements. You can use an API test to test a connection to a Web Service or database, the ability of your application to upload or download a specific file or string from the file system or information database, and similar application background services. Using the Toolbox pane, you drag various API service activities onto the canvas to create a test. After compiling the test, UFT runs each step individually and reports which steps were performed as well as which steps succeeded or failed.

The image below shows an API test canvas.



G2 Los Angeles

	A	B	C
1	CustomerName	FlightNumber	NumberOf
2	John Freeman	1042	
3	Jack Dowson	1161	
4	Ann Frank	1089	
5	Kate Sterling	17079	

Input / Output / Sheet3

For details, see "UFT API Testing Overview" on page 1590.

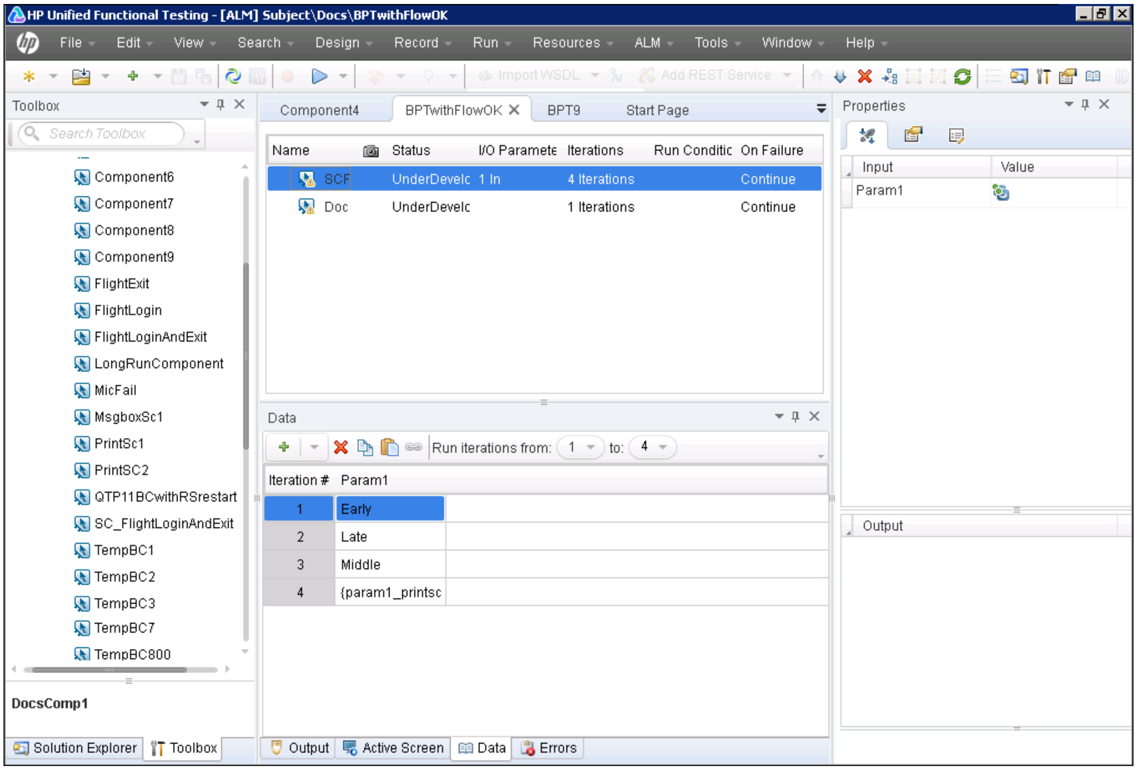
Business Process Tests at a Glance

Relevant for: business process tests

Business process tests enable you to create a scenario that contains a sequence of business components or flows to test a specific business process in your application. You can group flows and/or components together to run them for a specific number of iterations, and to use parameters for an entire group.

> **Note:** The first time that you open a business process test or flow from UFT, you must connect to ALM as a user with administrator privileges on the computer on which you are connecting.

The image below shows a business process test, as well as the components and flows available to the test, iteration data for the selected component in the test, and test properties.



Name	Status	I/O Paramete	Iterations	Run Conditic	On Failure
SCF	UnderDevelc	1 In	4 Iterations		Continue
Doc	UnderDevelc		1 Iterations		Continue

Run iterations from: 1 to: 4

Iteration #	Param1
1	Early
2	Late
3	Middle
4	{param1_printsc

Input	Value
Param1	

For details, see "Business Process Testing in UFT" on page 2062.

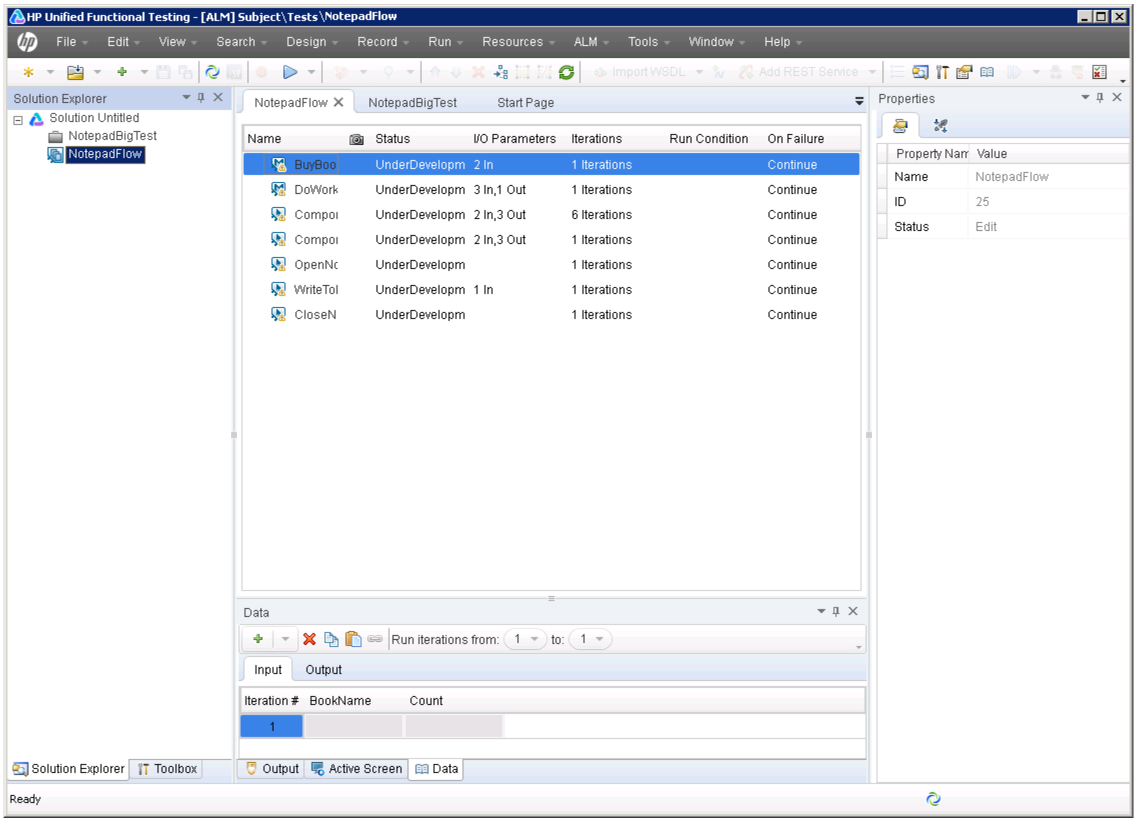
Business Process Testing Flows at a Glance

Relevant for: business process flows

Business process flows enable you create a logical set of business components in a fixed sequence to perform a specific business task, and can be added to business process tests to run as a unit.

> **Note:** The first time that you open a business process test or flow from UFT, you must connect to ALM as a user with administrator privileges on the computer on which you are connecting.

The image below shows a business process flow selected in the Solution Explorer and displayed in the document pane, as well iteration data for the selected component in the test, and the flow's test properties.



Name	Status	I/O Parameters	Iterations	Run Condition	On Failure
BuyBoo	UnderDevelopm	2 In	1 Iterations		Continue
DoWork	UnderDevelopm	3 In,1 Out	1 Iterations		Continue
Compoı	UnderDevelopm	2 In,3 Out	6 Iterations		Continue
Compoı	UnderDevelopm	2 In,3 Out	1 Iterations		Continue
OpenNc	UnderDevelopm		1 Iterations		Continue
WriteTol	UnderDevelopm	1 In	1 Iterations		Continue
CloseN	UnderDevelopm		1 Iterations		Continue

Property Nam	Value
Name	NotepadFlow
ID	25
Status	Edit

Run iterations from: 1 to: 1

Input Output

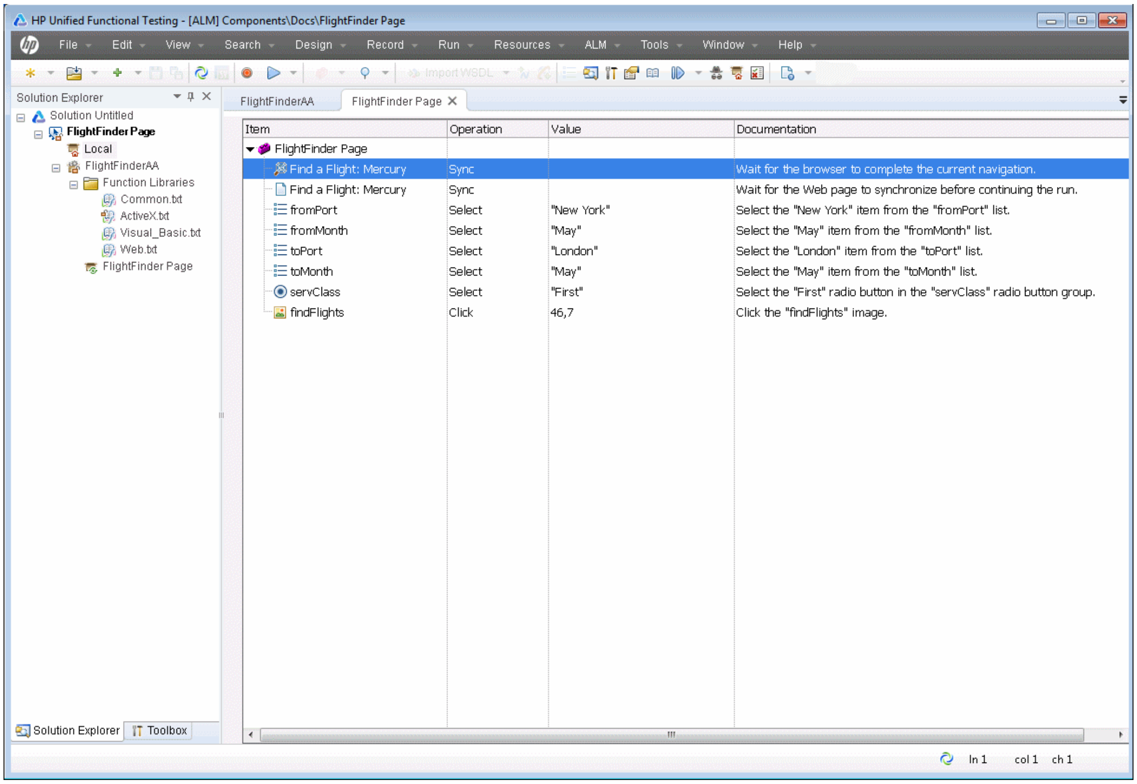
Iteration #	BookName	Count
1		

For details, see "Business Process Testing in UFT" on page 2062.

Keyword GUI Components at a Glance

Relevant for: Keyword GUI components only

Keyword GUI components enable you to test a specific area of your application using a keyword-driven format in the Keyword View. Each component consists of steps divided in a modular format that are automatically documented as they are completed. You can also add user-defined functions to your keyword component to specify a specific function or operation to perform. Keyword GUI components are saved with your ALM project, making them available for other testing projects and resources.



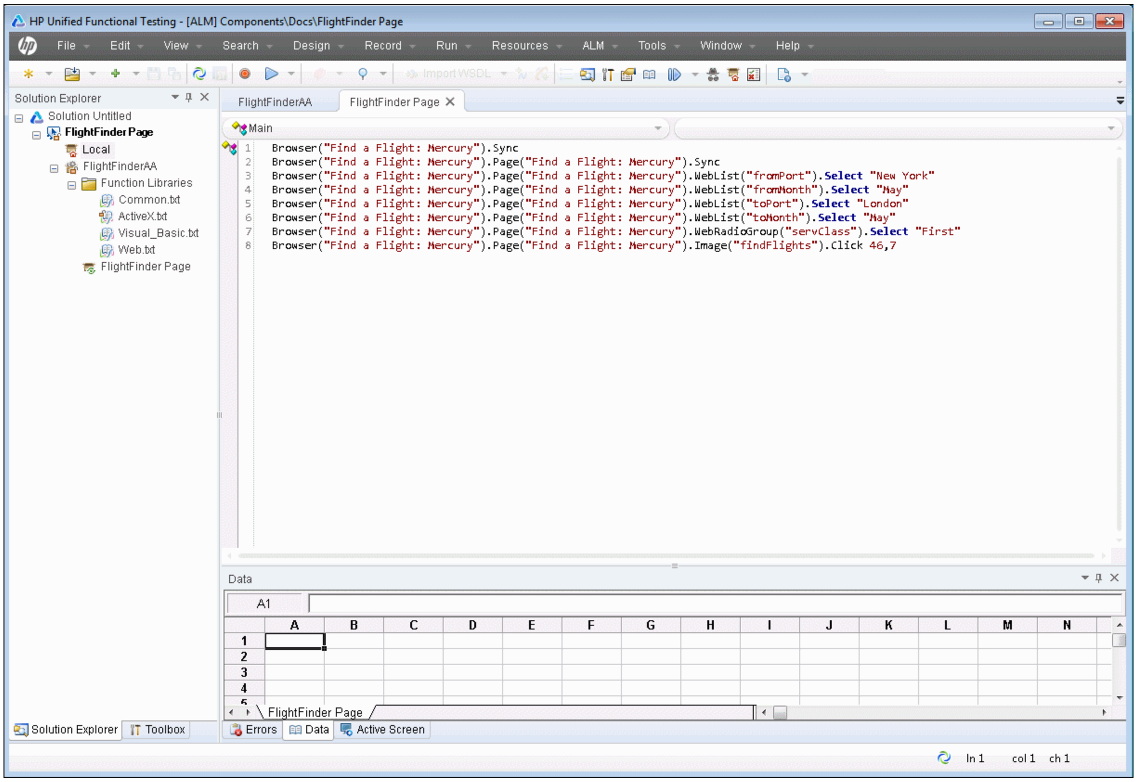
Item	Operation	Value	Documentation
FlightFinder Page			
Find a Flight: Mercury	Sync		Wait for the browser to complete the current navigation.
Find a Flight: Mercury	Sync		Wait for the Web page to synchronize before continuing the run.
fromPort	Select	"New York"	Select the "New York" item from the "fromPort" list.
fromMonth	Select	"May"	Select the "May" item from the "fromMonth" list.
toPort	Select	"London"	Select the "London" item from the "toPort" list.
toMonth	Select	"May"	Select the "May" item from the "toMonth" list.
servClass	Select	"First"	Select the "First" radio button in the "servClass" radio button group.
findFlights	Click	46,7	Click the "findFlights" image.

For details, see "Keyword GUI Components Overview" on page 2084.

Scripted GUI Components at a Glance

Relevant for: Scripted GUI components only

Scripted GUI components enable you to create components using the full functionality of both the Keyword View and Editor and the tools associated with these views. Scripted GUI components also enable you to use more complex functionality into your component through the addition of VBScript and other programming statement steps in addition to keyword-driven steps. Scripted GUI components are saved with your ALM project, making them available for other testing projects and resources.



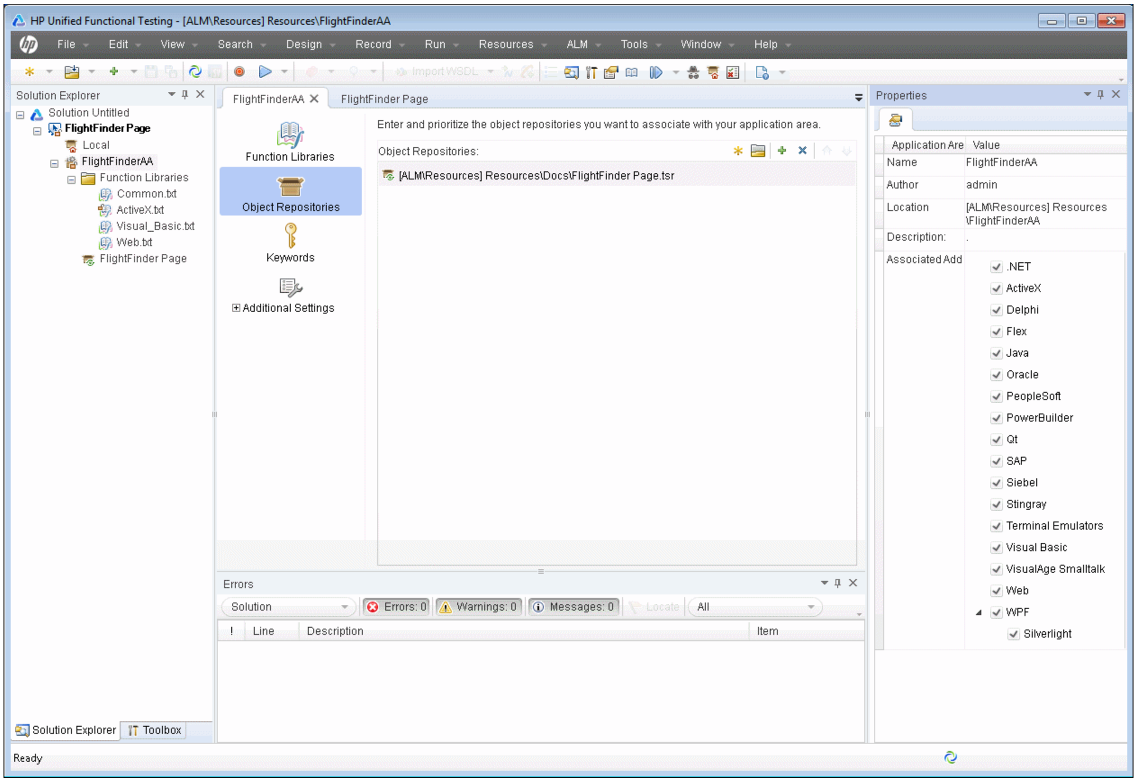
```
Browser("Find a Flight: Mercury").Sync
Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").Sync
Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("fromPort").Select "New York"
Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("fromMonth").Select "May"
Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("toPort").Select "London"
Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("toMonth").Select "May"
Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebRadioGroup("servClass").Select "First"
Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").Image("findFlights").Click 46,7
```

For details, see "Scripted GUI Components Overview" on page 2085.

Application Areas at a Glance

Relevant for: GUI components only

Each business component is based on an application area that provides it with settings and links to specific resource files, such as function library files, shared object repositories (that contain the test objects used by the application), and recovery scenario files. You define most of these assets in the application area. Associated add-ins and other general settings are defined in the Properties pane.



Enter and prioritize the object repositories you want to associate with your application area.

Object Repositories:

[ALM\Resources] Resources\Docs\FlightFinder Page.tsr

Application Are	Value
Name	FlightFinderAA
Author	admin
Location	[ALM\Resources] Resources \FlightFinderAA
Description:	.
Associated Add	✓ .NET ✓ ActiveX ✓ Delphi ✓ Flex ✓ Java ✓ Oracle ✓ PeopleSoft ✓ PowerBuilder ✓ Qt ✓ SAP ✓ Siebel ✓ Stingray ✓ Terminal Emulators ✓ Visual Basic ✓ VisualAge Smalltalk ✓ Web ✓ WPF ✓ Silverlight

The application area contains a number of panes that are accessed by the buttons in the left sidebar:

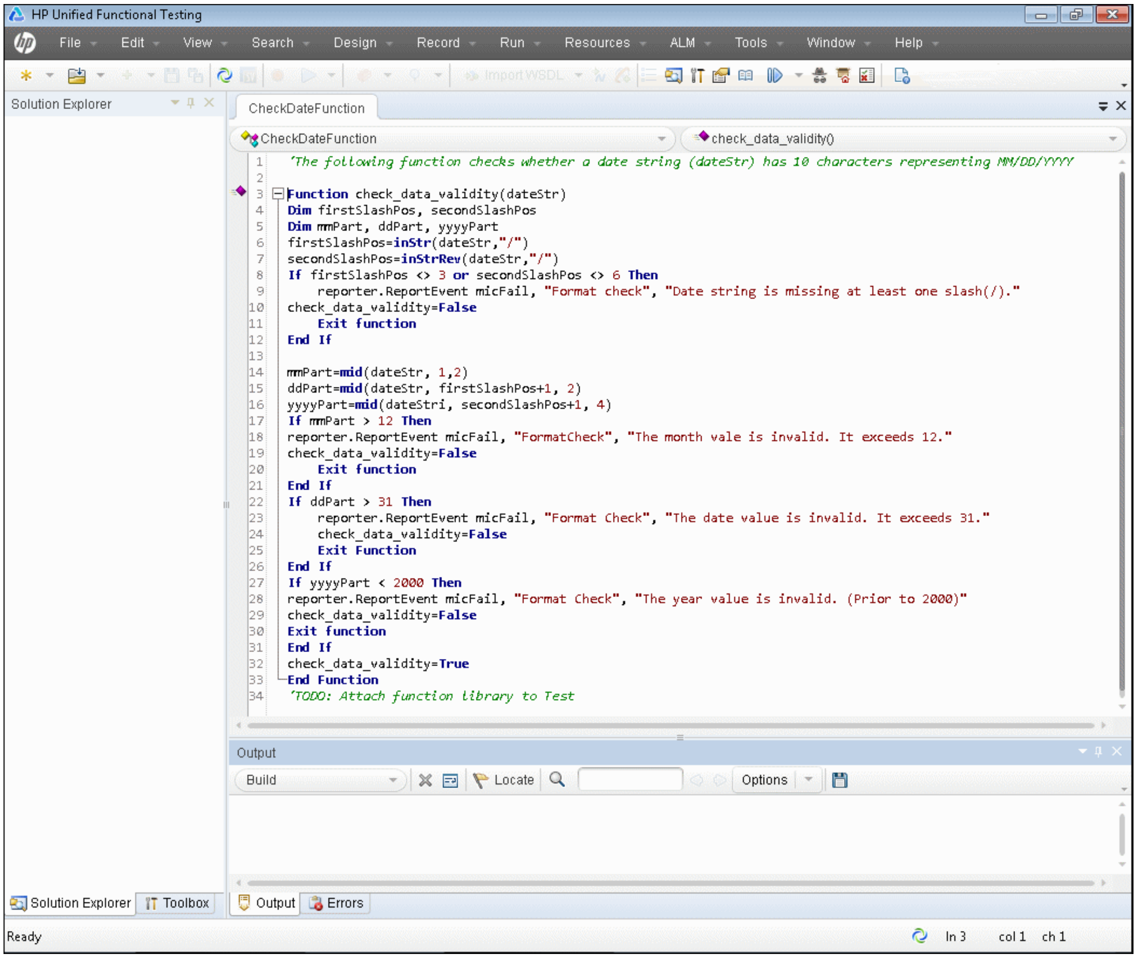
Pane	Description
Function Libraries	Displays the associated function libraries and enables you to associate additional function libraries with this application area and to prioritize them.
Object Repositories	Displays the associated object repositories and enables you to associate shared object repositories with this application area and to prioritize them.
Keywords	Enables you to set the keywords that are available to this application area and to view their individual properties.
Additional Settings	Enables you to specify settings for the applications used in designing an application area, recovery scenarios for the application area, and log tracking and system monitoring preferences.

For details, see "Application Areas" on page 2095 and "General Properties Tab (Properties Pane for Application Areas)" on page 391

Function Libraries at a Glance

Relevant for: GUI tests and components

Function libraries enable you to create user-defined functions that you can associate with and can call in a test or component. Each function library is a separate document containing VBscript functions, subroutines, classes, modules, and so on. Function libraries use the same editing and debugging features that are available in the Editor.



```
 'The following function checks whether a date string (dateStr) has 10 characters representing MM/DD/YYYY

Function check_data_validity(dateStr)
Dim firstSlashPos, secondSlashPos
Dim mmPart, ddPart, yyyyPart
firstSlashPos=inStr(dateStr,"/")
secondSlashPos=inStrRev(dateStr,"/")
If firstSlashPos <> 3 or secondSlashPos <> 6 Then
    reporter.ReportEvent micFail, "Format check", "Date string is missing at least one slash(/)."
check_data_validity=False
    Exit function
End If

mmPart=mid(dateStr, 1,2)
ddPart=mid(dateStr, firstSlashPos+1, 2)
yyyyPart=mid(dateStri, secondSlashPos+1, 4)
If mmPart > 12 Then
reporter.ReportEvent micFail, "FormatCheck", "The month vale is invalid. It exceeds 12."
check_data_validity=False
    Exit function
End If
If ddPart > 31 Then
    reporter.ReportEvent micFail, "Format Check", "The date value is invalid. It exceeds 31."
    check_data_validity=False
    Exit Function
End If
If yyyyPart < 2000 Then
reporter.ReportEvent micFail, "Format Check", "The year value is invalid. (Prior to 2000)"
check_data_validity=False
Exit function
End If
check_data_validity=True
End Function
 'TODO: Attach function library to Test
```

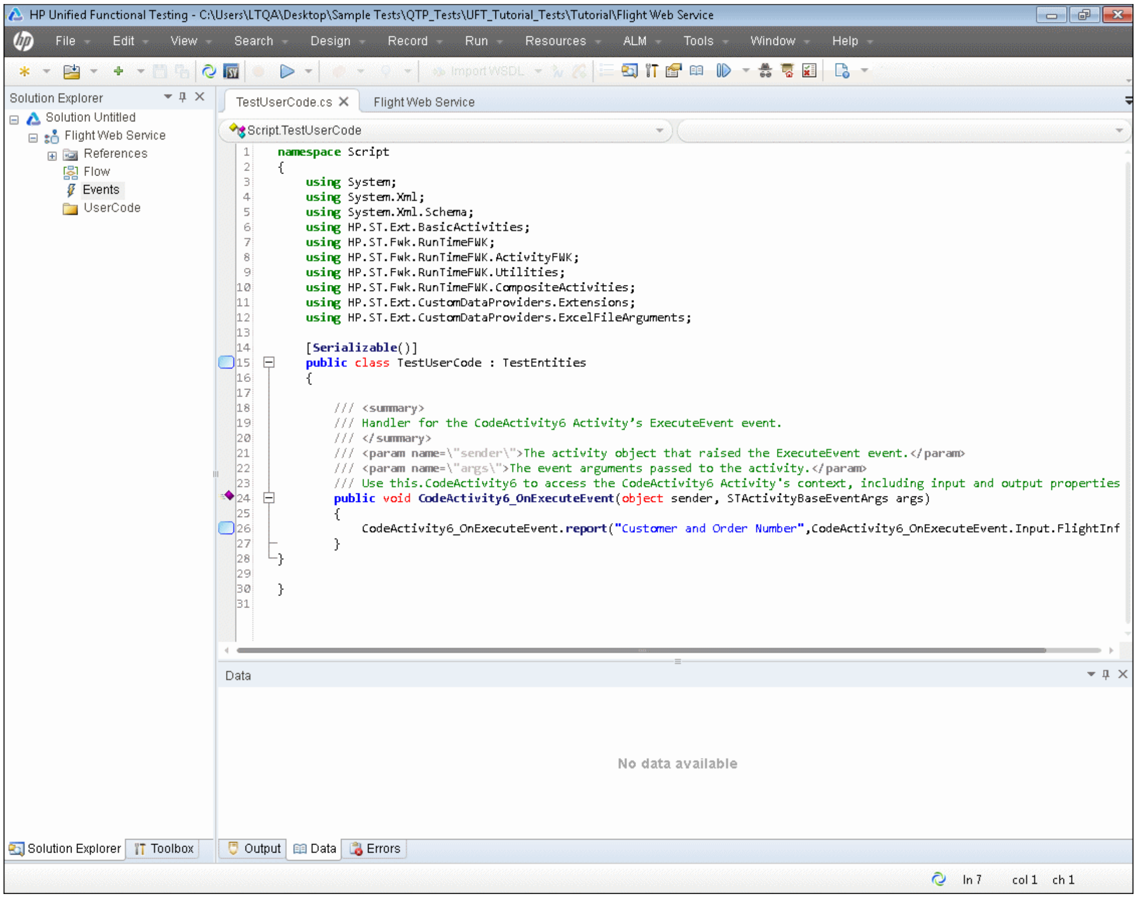
For details, see "User-Defined Functions and Function Libraries" on page 1029.

User Code Files at a Glance

Relevant for: API testing only

User code files enable you to create user-defined events (in the TestUserCode.cs file) to run as part of your test steps or user-defined classes that a test calls during a run session. The

TestUserCode.cs file contains all events created as part of a test. Other user code files are separate documents containing all the code needed during the test session.



```
namespace Script
{
    using System;
    using System.Xml;
    using System.Xml.Schema;
    using HP.ST.Ext.BasicActivities;
    using HP.ST.Fwk.RunTimeFWK;
    using HP.ST.Fwk.RunTimeFWK.ActivityFWK;
    using HP.ST.Fwk.RunTimeFWK.Utilities;
    using HP.ST.Fwk.RunTimeFWK.CompositeActivities;
    using HP.ST.Ext.CustomDataProviders.Extensions;
    using HP.ST.Ext.CustomDataProviders.ExcelFileArguments;

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity’s ExecuteEvent event.
        /// </summary>
        /// <param name=\"sender\">The activity object that raised the ExecuteEvent event.</param>
        /// <param name=\"args\">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, including input and output properties
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs args)
        {
            CodeActivity6_OnExecuteEvent.report("Customer and Order Number",CodeActivity6_OnExecuteEvent.Input.FlightInf
        }
    }

}
```

For details, see "Writing Code for API Test Events - Overview" on page 1938.

UFT Panes at a Glance

Relevant for: GUI tests and components and API testing

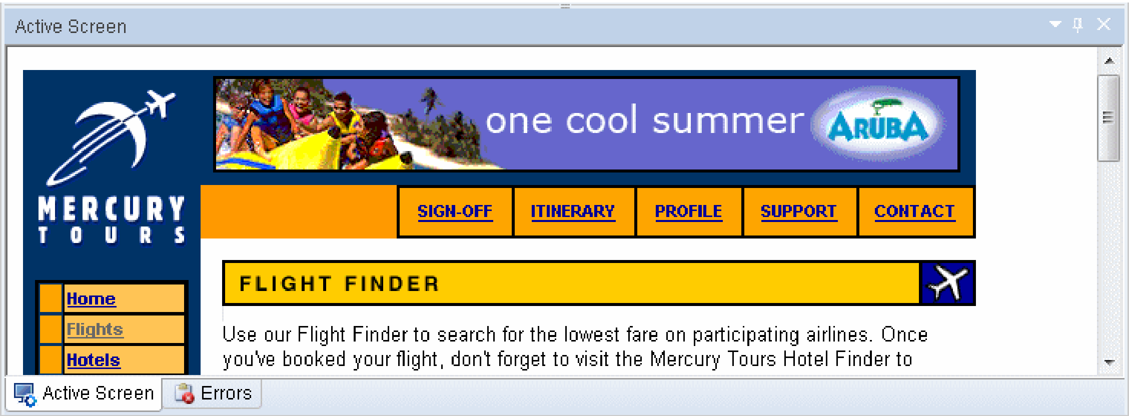
The following sections describe the available UFT panes:

Active Screen Pane at a Glance

Relevant for: GUI tests and scripted GUI components

The Active Screen provides a snapshot of your application as it appeared when you performed a certain step during a recording session. Additionally, depending on the Active Screen capture options that you used while recording, the page displayed in the Active Screen can contain detailed property information about each object displayed on the page.

To view the Active Screen, select **View > Active Screen**.



Use our Flight Finder to search for the lowest fare on participating airlines. Once you've booked your flight, don't forget to visit the Mercury Tours Hotel Finder to

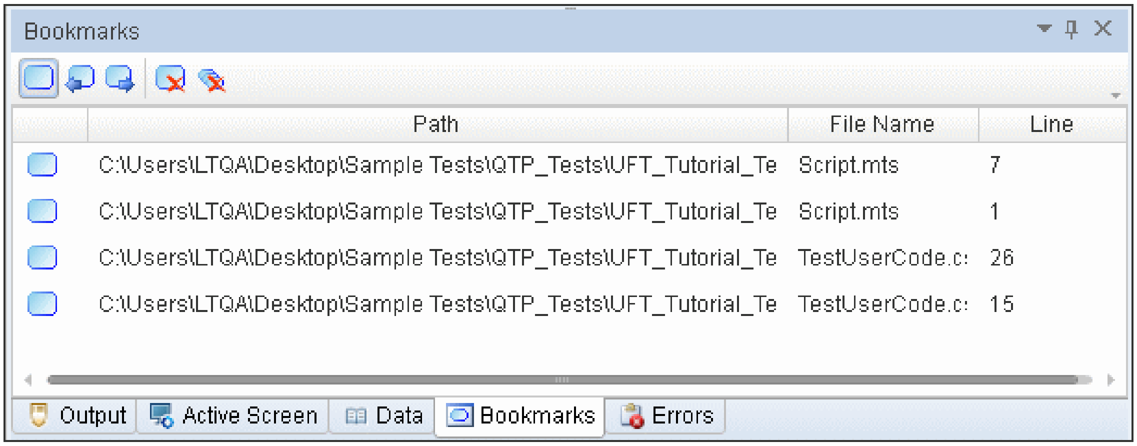
For details, see "Active Screen Pane" on page 193.

Bookmarks Pane at a Glance

Relevant for: GUI tests scripted GUI components, function libraries, and user code files

The Bookmarks pane displays all bookmarks inserted into your tests, component, function libraries, or user code files. The pane enables you to create bookmarks, view the information about inserted bookmarks, and navigate to the document containing the bookmark.

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



	Path	File Name	Line
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	Script.mts	7
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	Script.mts	1
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	TestUserCode.c:	26
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	TestUserCode.c:	15

For details, see "Bookmarks Overview" on page 204.

Data Pane at a Glance

Relevant for: GUI tests and components, API testing, and business process testing

The Data pane displays the relevant data for your test or component, or business process testing flow. It's a spreadsheet-like table with columns and rows representing the data applicable to your document, and assists you in parameterizing that data.

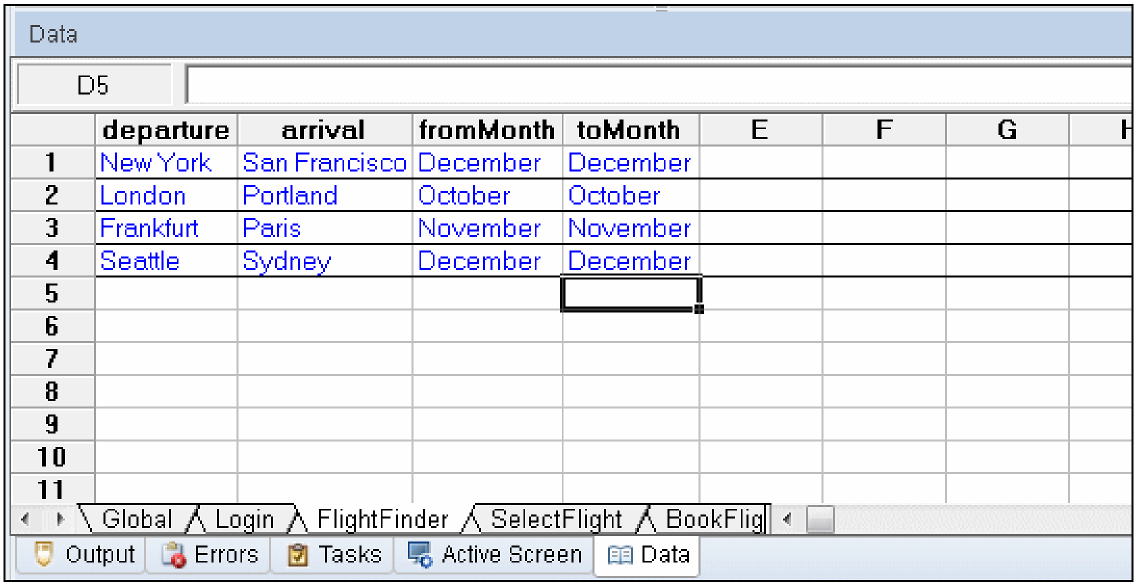
For GUI testing: The Data displays a Global tab showing the data relevant to the entire test, and separate tabs for each action contained within the test.

For API testing: The Data pane can display data from a local data table, an Excel spreadsheet, a database connection, or an XML file.

For business process testing: The Data pane displays iteration data related to the selected component, enables you to link iteration parameter data to other parameter values in your test or flow, as well as promote component parameters to test or flow parameters.

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

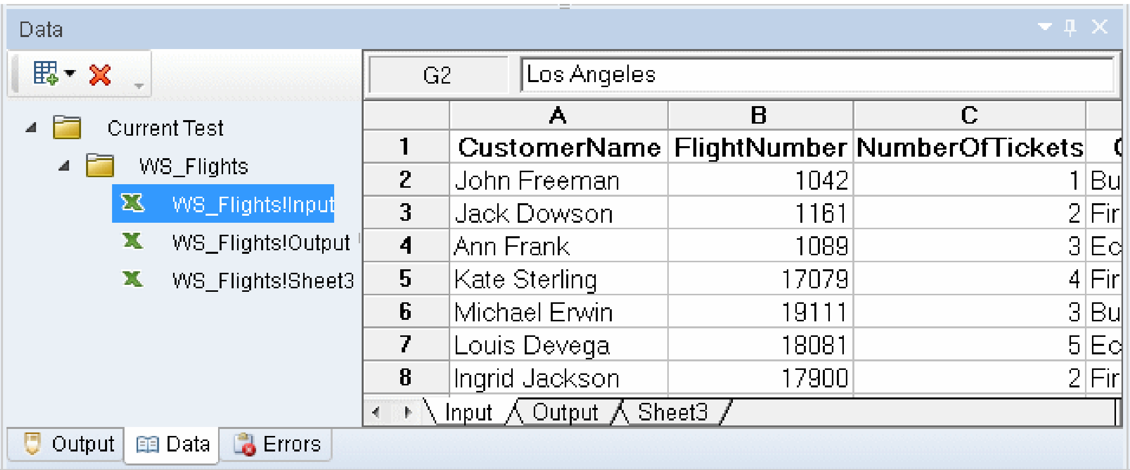
The image below shows the Data pane for GUI testing.



D5

	departure	arrival	fromMonth	toMonth	E	F	G	H
1	New York	San Francisco	December	December				
2	London	Portland	October	October				
3	Frankfurt	Paris	November	November				
4	Seattle	Sydney	December	December				
5								
6								
7								
8								
9								
10								
11								

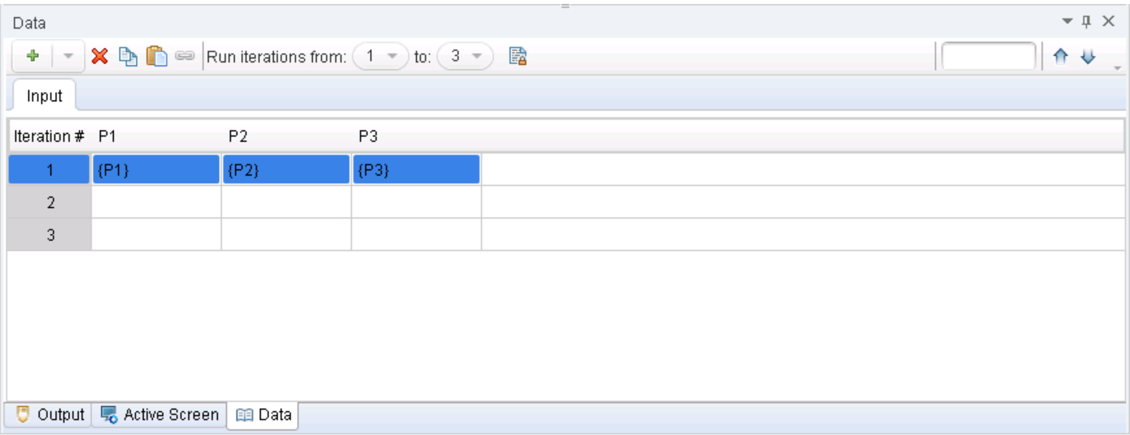
The image below shows the Data pane for API testing.



G2 Los Angeles

	A	B	C	
1	CustomerName	FlightNumber	NumberOfTickets	C
2	John Freeman	1042	1	Bu
3	Jack Dowson	1161	2	Fir
4	Ann Frank	1089	3	Ec
5	Kate Sterling	17079	4	Fir
6	Michael Erwin	19111	3	Bu
7	Louis Devega	18081	5	Ec
8	Ingrid Jackson	17900	2	Fir

The image below shows the Data pane for business process testing.



Run iterations from: 1 to: 3

Iteration #	P1	P2	P3
1	{P1}	{P2}	{P3}
2			
3			

For details about the Data pane, see "Data Pane Overview" on page 222.

Errors Pane at a Glance

Relevant for: GUI tests and components and API testing

The Errors pane provides a list of missing resources and coding syntax errors from your test. If the pane is not currently displayed, UFT automatically opens it when an error is detected. You can double-click an error to locate in the error in your test or component.

Missing resources are the resources that are specified in your test or component but cannot be found. Information errors provide a list of coding syntax errors in your test or function library scripts.

For GUI testing: Missing resources for GUI tests or components can include calls to missing actions, missing function libraries, missing recovery scenarios, missing environment variable XML files, unmapped shared object repositories, and parameters that are connected to shared object repositories. Each time you open your test, component, or application area, UFT automatically checks that all specified resources are accessible. If it finds any resources that are not accessible, UFT lists them in the Errors pane.

In addition, when you create VBScript for your test or components steps, UFT automatically checks for VBScript syntax errors in your script, and shows them in the Errors pane.

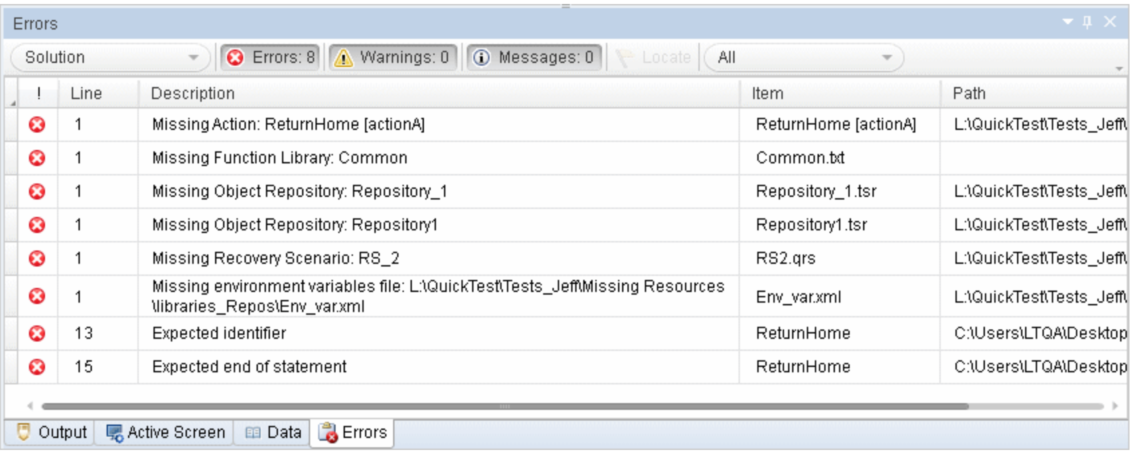
For API testing: Missing resources listed in this pane are missing resource files, such as references called by the test during a run session or user functions or objects called by a test.

The pane also displays coding syntax errors or individual step property errors. If there is a coding syntax error in a user code file or a property value error in a step, it is displayed in the Errors pane.

In addition, if a property value for a step is undefined, it is listed in the Errors pane as a test error.

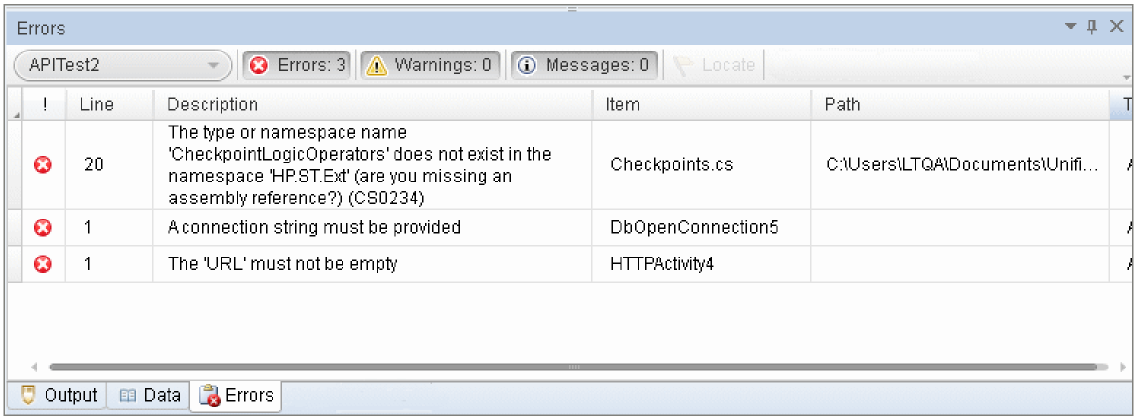
To view the Errors pane, select **View > Errors**.

The image below shows the Errors pane for GUI testing



!	Line	Description	Item	Path
⊗	1	Missing Action: ReturnHome [actionA]	ReturnHome [actionA]	L:\QuickTest\Tests_Jeff\
⊗	1	Missing Function Library: Common	Common.txt	
⊗	1	Missing Object Repository: Repository_1	Repository_1.tsr	L:\QuickTest\Tests_Jeff\
⊗	1	Missing Object Repository: Repository1	Repository1.tsr	L:\QuickTest\Tests_Jeff\
⊗	1	Missing Recovery Scenario: RS_2	RS2.qrs	L:\QuickTest\Tests_Jeff\
⊗	1	Missing environment variables file: L:\QuickTest\Tests_Jeff\Missing Resources \libraries_Repos\Env_var.xml	Env_var.xml	L:\QuickTest\Tests_Jeff\
⊗	13	Expected identifier	ReturnHome	C:\Users\LTQA\Desktop
⊗	15	Expected end of statement	ReturnHome	C:\Users\LTQA\Desktop

The image below shows the Errors pane for API testing



!	Line	Description	Item	Path
☒	20	The type or namespace name 'CheckpointLogicOperators' does not exist in the namespace 'HP.ST.Ext' (are you missing an assembly reference?) (CS0234)	Checkpoints.cs	C:\Users\LTQA\Documents\Unifi...
☒	1	A connection string must be provided	DbOpenConnection5	
☒	1	The 'URL' must not be empty	HTTPActivity4	

For details, see "Errors Pane" on page 364.

Debug Panes at a Glance

Relevant for: GUI tests, scripted GUI components, function libraries and user code files

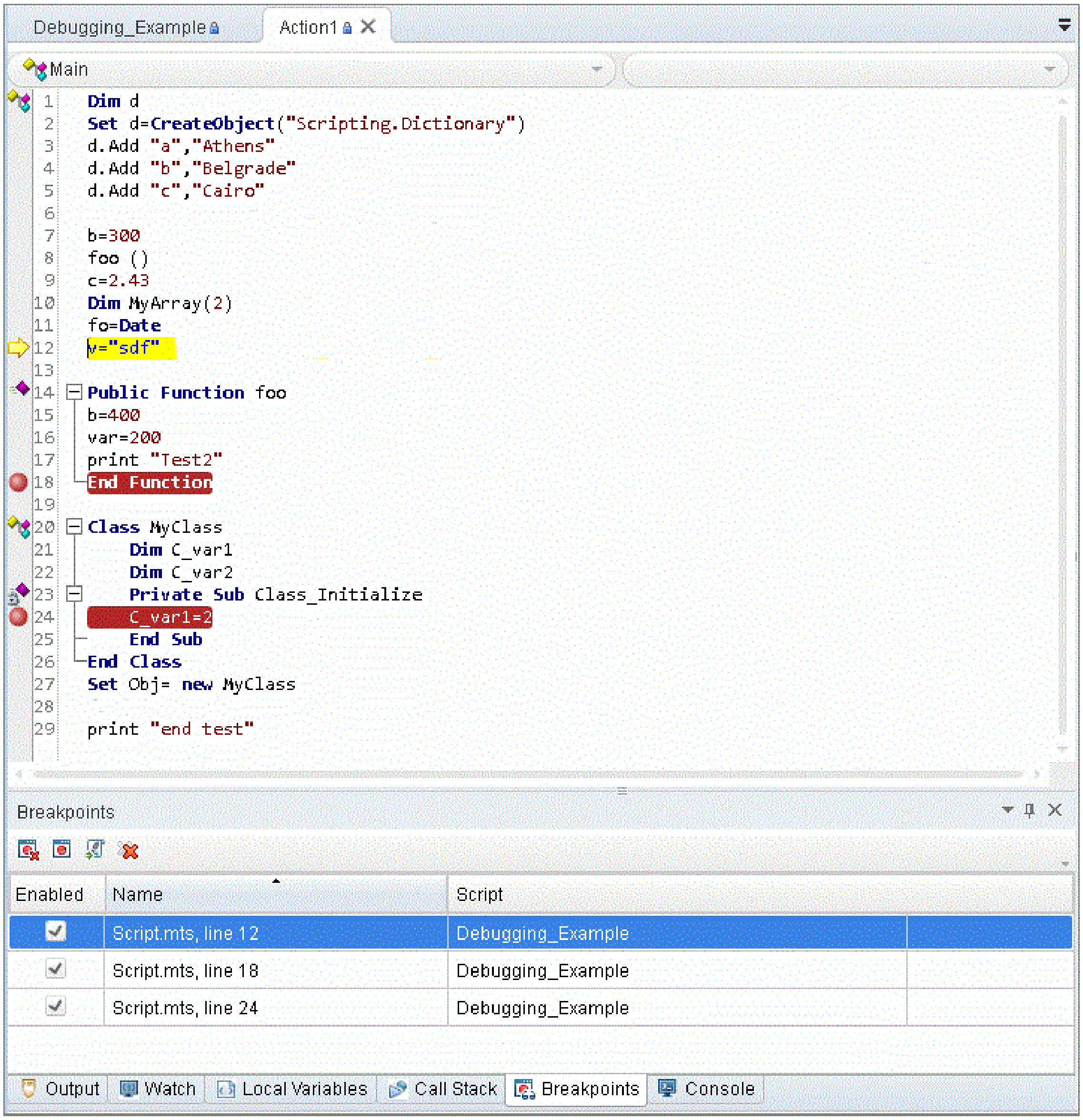
The Debug panes assist you in debugging your document or user code file, and contains the following options:

Tab	Description
Breakpoints	Displays information about breakpoints inserted into tests, function libraries, or user code files and allows you enable or disable the breakpoints. For details, see " Breakpoints Pane" on page 277.
Call Stack	Displays information about the function and method calls currently running in a test, function library, or user code file. For details, see "Call Stack Pane" on page 281.
Loaded Modules (**API testing only**)	Displays information on the .dll files associated with the current run session. For details, see "Loaded Modules Pane (API Testing) " on page 285.
Threads (**API testing only**)	Displays information about all the threads running in the current context of the test. For details, see "Threads Pane (API Testing) " on page 287.
Local Variables	Displays the current value and type of all variables that were recognized up to the last step performed during the run session that you are debugging. You can also set or modify the values of the variables that are displayed. For details, see "Local Variables Pane" on page 288.

Tab	Description
Console	Enables you to run lines of script to set or modify the current value of a variable, code object, property, method, or function call in your test or function library. For details, see "Console Pane" on page 292.
Watch	Displays the current value and type of any variable or expression that you added to the Watch pane. For details, see "Watch Pane" on page 297.

To view the Debug panes, select **View > Debug** and select the specific Debug pane you want to view.

This image below shows the Breakpoints pane for a **GUI test** visible with the other panes open as individual tabs. You can display any Debug pane by clicking on the relevant tab.

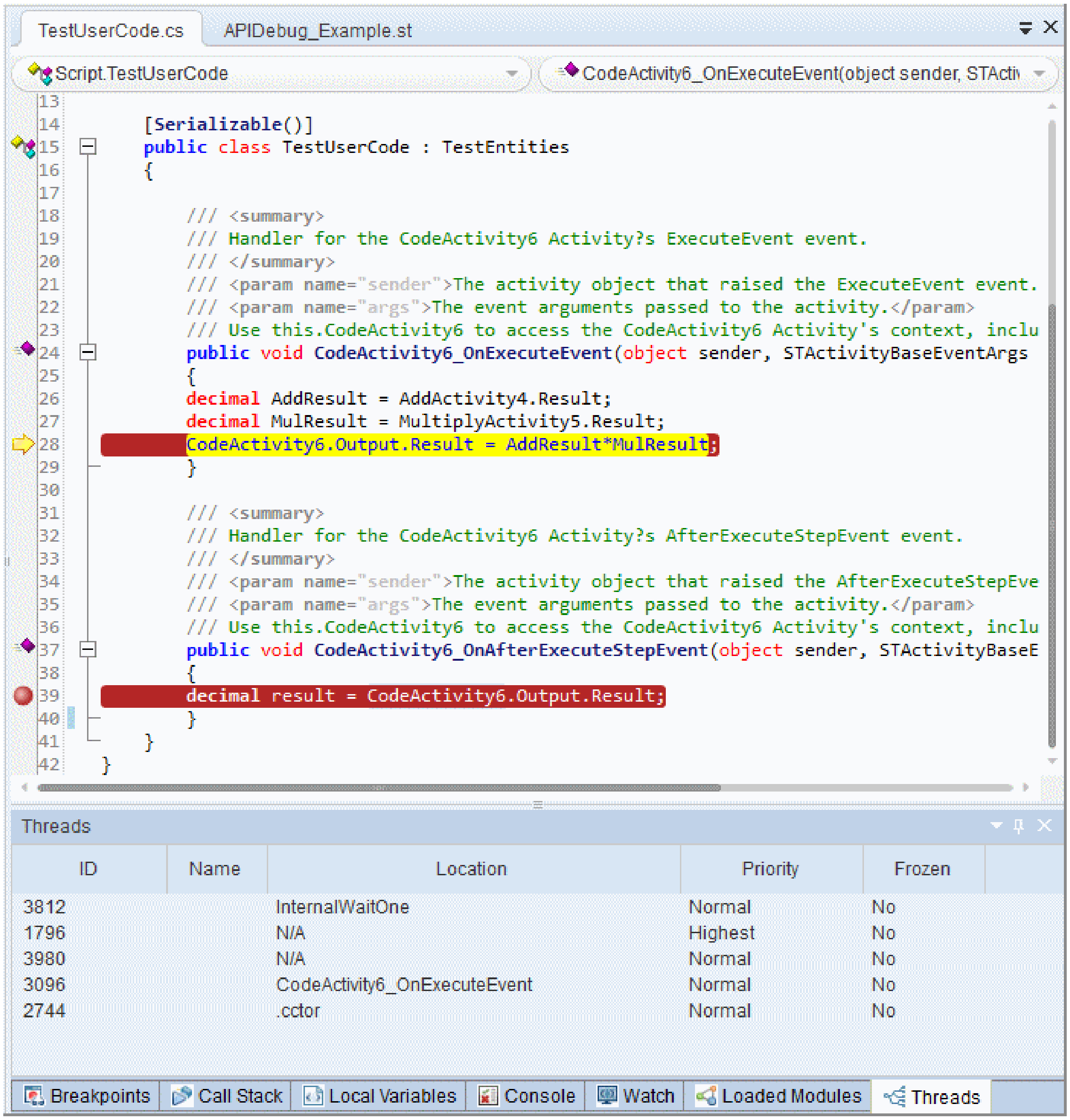


```
1   Dim d
2   Set d=CreateObject("Scripting.Dictionary")
3   d.Add "a","Athens"
4   d.Add "b","Belgrade"
5   d.Add "c","Cairo"
6
7   b=300
8   foo ()
9   c=2.43
10  Dim MyArray(2)
11  fo=Date
12  v="sdf"
13
14  Public Function foo
15  b=400
16  var=200
17  print "Test2"
18  End Function
19
20  Class MyClass
21      Dim C_var1
22      Dim C_var2
23      Private Sub Class_Initialize
24      C_var1=2
25      End Sub
26  End Class
27  Set Obj= new MyClass
28
29  print "end test"
```

Breakpoints

Enabled	Name	Script
☑	Script.mts, line 12	Debugging_Example
☑	Script.mts, line 18	Debugging_Example
☑	Script.mts, line 24	Debugging_Example

This image below shows the Threads pane for an **API test** visible with the other panes open as individual tabs. You can display any debug pane by clicking on the relevant tab.



```

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

Threads

ID	Name	Location	Priority	Frozen
3812		InternalWaitOne	Normal	No
1796		N/A	Highest	No
3980		N/A	Normal	No
3096		CodeActivity6_OnExecuteEvent	Normal	No
2744		.cctor	Normal	No

For details, see "Debug Panes" on page 271.

Output Pane at a Glance

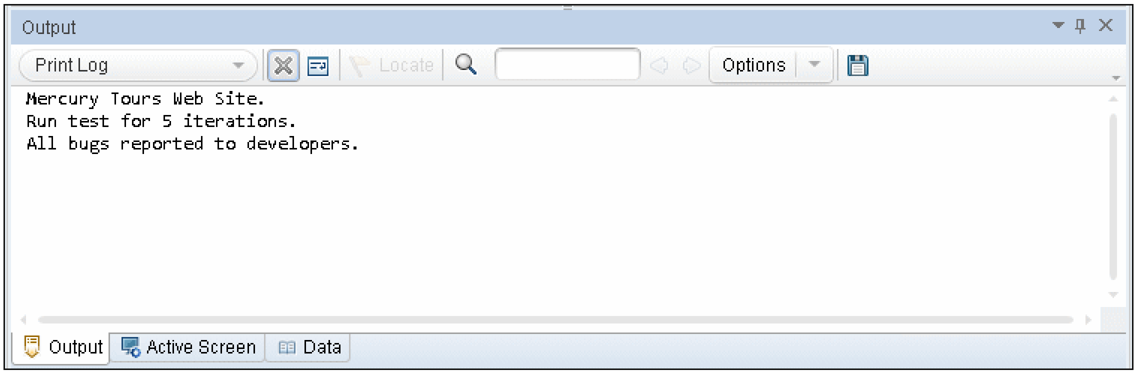
Relevant for: GUI tests and components and API testing

For GUI tests: This pane enables you to view information that is sent using the **Print** Utility statement during the run session.

For API testing: This pane enables you to view the Output log for the compilation and test run.

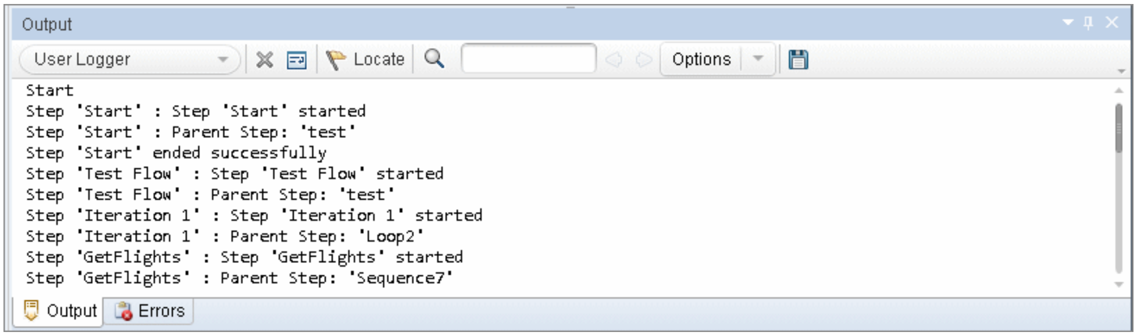
To view the Output pane, select **View > Output**.

The image below shows the Output pane for **GUI testing**.



```
Mercury Tours Web Site.
Run test for 5 iterations.
All bugs reported to developers.
```

The image below shows the Output pane for .**API testing**



```
Start
Step 'Start' : Step 'Start' started
Step 'Start' : Parent Step: 'test'
Step 'Start' ended successfully
Step 'Test Flow' : Step 'Test Flow' started
Step 'Test Flow' : Parent Step: 'test'
Step 'Iteration 1' : Step 'Iteration 1' started
Step 'Iteration 1' : Parent Step: 'Loop2'
Step 'GetFlights' : Step 'GetFlights' started
Step 'GetFlights' : Parent Step: 'Sequence7'
```

For details, see "Output Pane" on page 379.

Properties Pane at a Glance

Relevant for: GUI tests and components, API testing, and business process testing

The Properties pane displays and enables you to edit properties and parameters for a selected test, action, component, function library, or application area. The information displayed in this pane is dependent upon the active document.

Additionally:

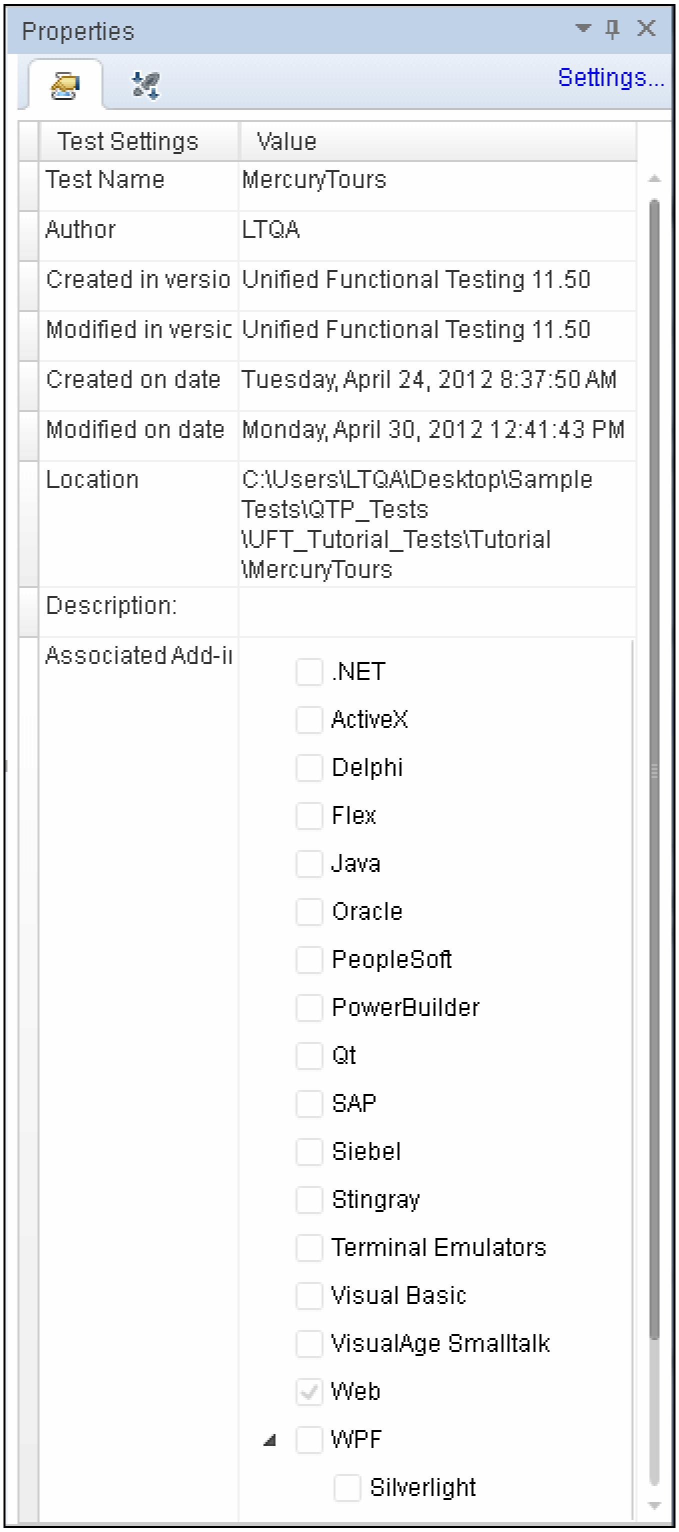
For GUI testing: The Properties pane enables you to view details about your test and specify input or output parameters for an action or test in the **Parameters** tab.

For API testing: The Properties pane enables you view and edit the property values, schemas, and event handlers for the different activities in the test canvas. Using the **Input/Checkpoints** tab, you can specify input and output values for the highlighted activity and set checkpoints for that activity. Using the **Events** tab, you can open the TestUserCode.cs file in the Editor, enabling you to write event code to use during a particular test step. Using the **Data Sources** tab (visible only when the test flow is selected), you can select data sources to use with your test.

For BPT in UFT: The Properties pane enables you to specify parameters and other details for your test, flow, or component.

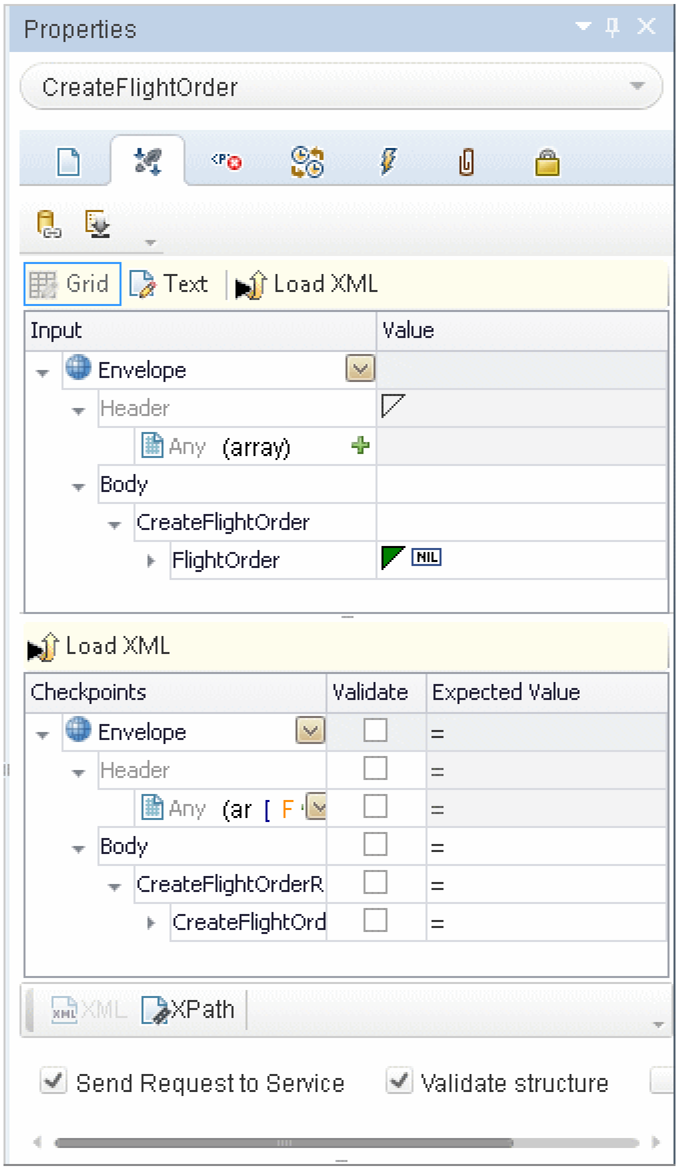
To view the Properties pane, select **View > Properties**.

The image below shows the Properties tab in the Properties pane for **GUI testing**. You can display the other tabs by clicking on them.



Test Settings	Value
Test Name	MercuryTours
Author	LTQA
Created in versio	Unified Functional Testing 11.50
Modified in versic	Unified Functional Testing 11.50
Created on date	Tuesday, April 24, 2012 8:37:50 AM
Modified on date	Monday, April 30, 2012 12:41:43 PM
Location	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests \UFT_Tutorial_Tests\Tutorial \MercuryTours
Description:	
Associated Add-ii	□ .NET □ ActiveX □ Delphi □ Flex □ Java □ Oracle □ PeopleSoft □ PowerBuilder □ Qt □ SAP □ Siebel □ Stingray □ Terminal Emulators □ Visual Basic □ VisualAge Smalltalk ☑ Web ◢ □ WPF □ Silverlight

The image below shows the **Input/Checkpoints** view for an API test step. You can display the other tabs by clicking on them.

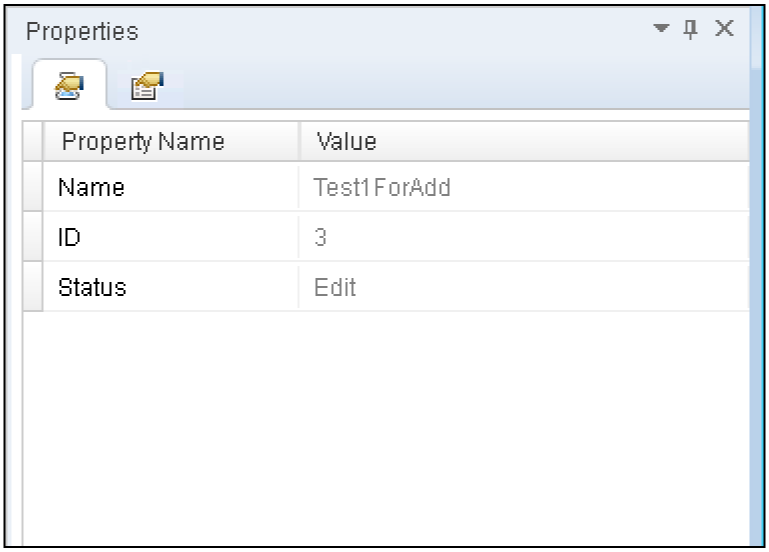


Input	Value
Envelope	
Header	
Any (array)	
Body	
CreateFlightOrder	
FlightOrder	NIL

Checkpoints	Validate	Expected Value
Envelope	□	=
Header	□	=
Any (ar [F	□	=
Body	□	=
CreateFlightOrderR	□	=
CreateFlightOrd	□	=

☑ Send Request to Service ☑ Validate structure

The image below shows the Properties tab in the Properties pane when a business process test is selected. You can display the **Component Parameters** and **Comments** tabs by selecting a component or flow in your test.



Property Name	Value
Name	Test1ForAdd
ID	3
Status	Edit

For details on the **GUI testing** Properties pane, see "Properties Pane User Interface (GUI Testing)" on page 387.

For details on the **API testing** Properties pane, see "Properties Pane User Interface (API Testing) " on page 402.

For details on the **BPT** Properties pane, see "Properties Pane User Interface (BPT in UFT)" on page 438.

Solution Explorer Pane at a Glance

Relevant for: GUI tests and components and API testing

The Solution Explorer Pane displays all the resources associated with a test or component. You can add, remove, and manage all of the resources in your test or to view and open all of the resources in your component through this pane. In addition, the Solution Explorer pane enables you to combine multiple types of tests, components, application areas, function libraries, and user code files into a single solution.

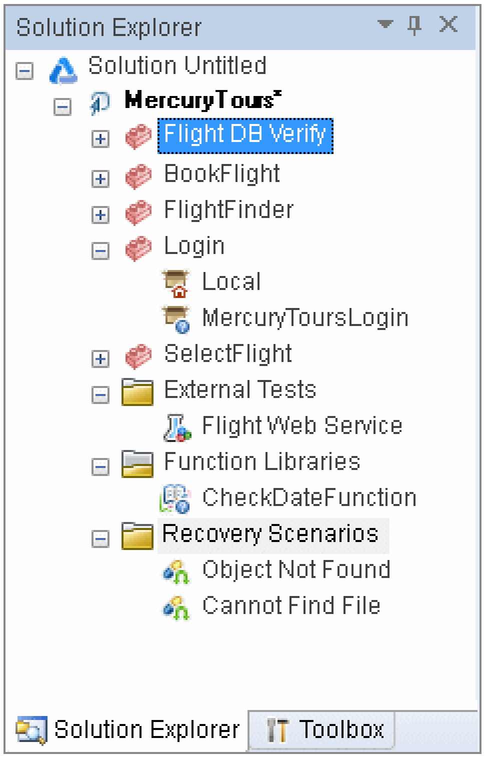
For GUI testing: GUI tests and actions or components are associated with resources such as function libraries, recovery scenarios, and object repositories (via an application area). The Solution Explorer pane displays each action of a GUI test as a node of the test, with its associated object repositories, application areas, function libraries, and recovery scenarios.

For API testing: APItest nodes can include numerous references, events, and actions. The Solution Explorer pane displays all external references for a test, the event handler code file (**TestUserCode.cs**), anyAPI-specific actions created as part of a test, and any user-defined code files added to a test.

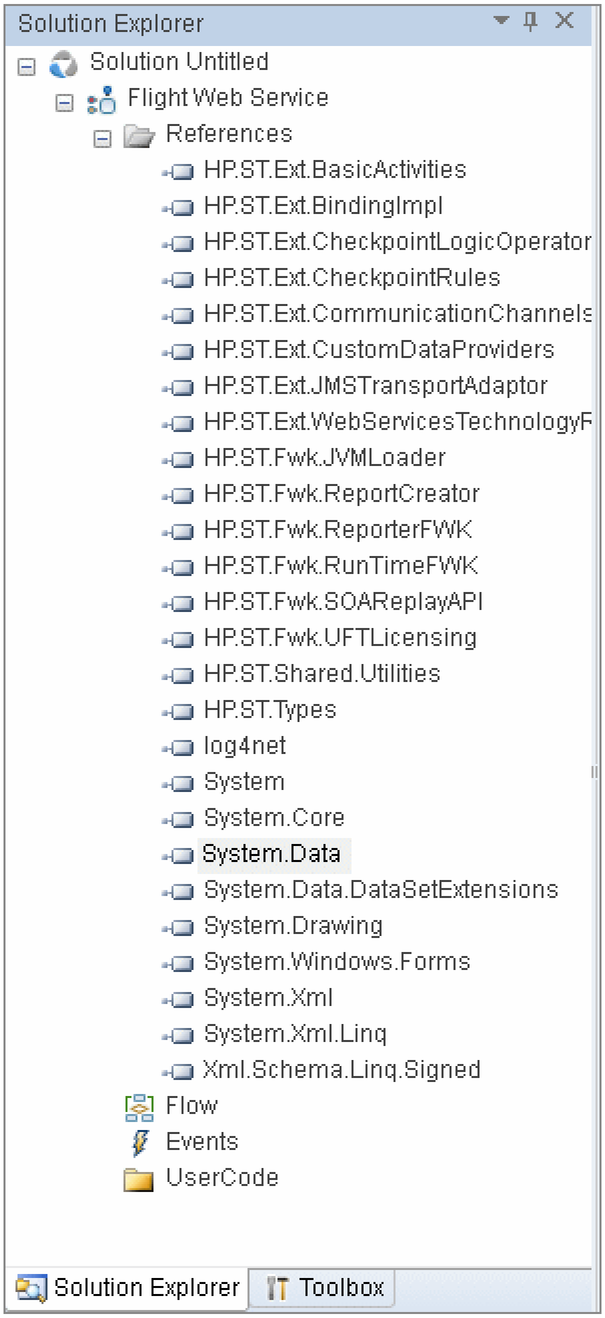
You can open most associated resources and references from the Solution Explorer pane by double-clicking on the name of the resource or reference.

To view the Solution Explorer pane, select **View > Solution Explorer**.

The image below shows the Solution Explorer pane for GUI testing.



The image below shows the Solution Explorer pane for **API testing.**



- Solution Untitled
  - Flight Web Service
    - References
      - HP.ST.Ext.BasicActivities
      - HP.ST.Ext.BindingImpl
      - HP.ST.Ext.CheckpointLogicOperator
      - HP.ST.Ext.CheckpointRules
      - HP.ST.Ext.CommunicationChannels
      - HP.ST.Ext.CustomDataProviders
      - HP.ST.Ext.JMSTransportAdaptor
      - HP.ST.Ext.WebServicesTechnologyF
      - HP.ST.Fwk.JVMLoader
      - HP.ST.Fwk.ReportCreator
      - HP.ST.Fwk.ReporterFWK
      - HP.ST.Fwk.RunTimeFWK
      - HP.ST.Fwk.SOAReplayAPI
      - HP.ST.Fwk.UFTLicensing
      - HP.ST.Shared.Utilities
      - HP.ST.Types
      - log4net
      - System
      - System.Core
      - System.Data
      - System.Data.DataSetExtensions
      - System.Drawing
      - System.Windows.Forms
      - System.Xml
      - System.Xml.Linq
      - Xml.Schema.Linq.Signed
    - Flow
    - Events
    - UserCode

For details, see "Solution Explorer Pane Overview" on page 471.

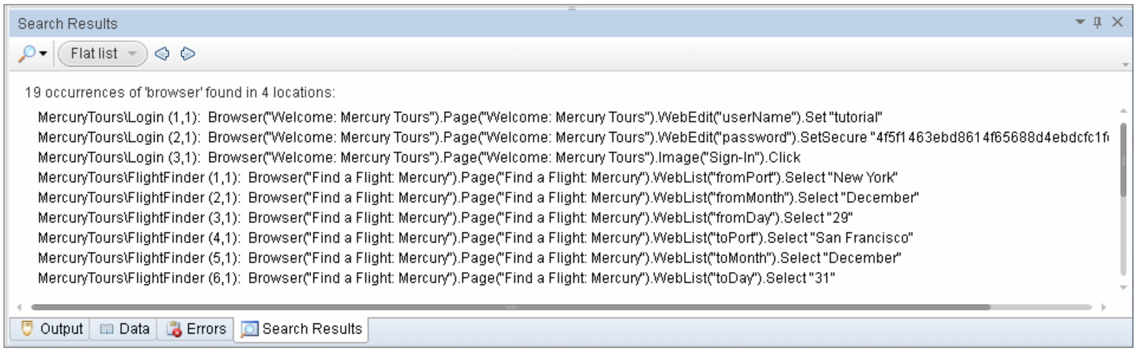
Search Results Pane at a Glance

Relevant for: GUI tests and components and API testing

The Search Results pane displays the results of searches performed using the options in the **Search** menu. Using this pane, you can browse the results of your search, locate a specific result, and perform recent searches in order to receive updated results.

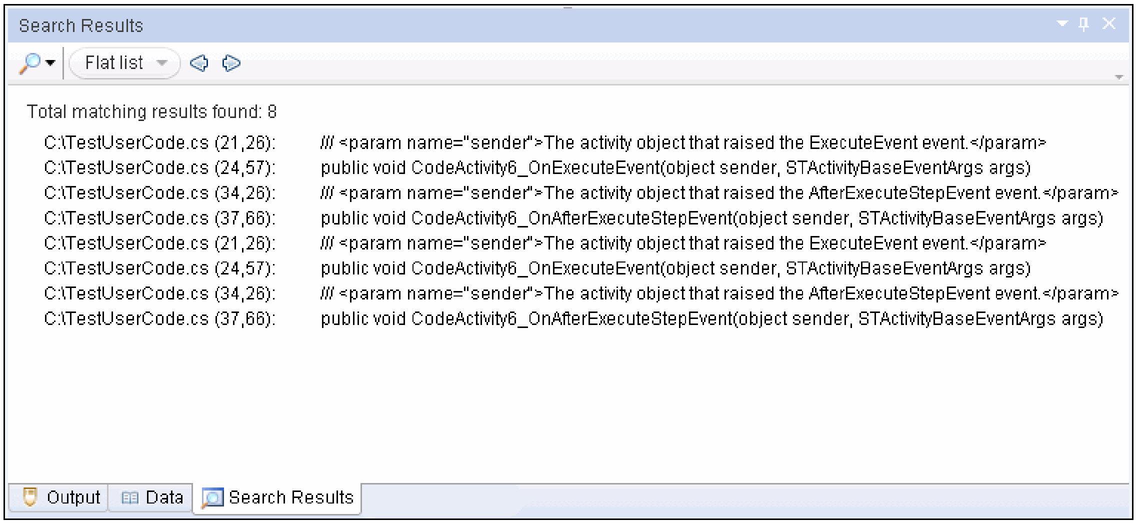
To view the Search Results pane, select **View > Search Results**.

The image below shows the Search Results pane for GUI testing.



```
19 occurrences of 'browser' found in 4 locations:
  MercuryTours\Login (1,1):  Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "tutorial"
  MercuryTours\Login (2,1):  Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetSecure "4f5f1463ebd8614f65688d4ebdcfc1f
  MercuryTours\Login (3,1):  Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click
  MercuryTours\FlightFinder (1,1):  Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("fromPort").Select "New York"
  MercuryTours\FlightFinder (2,1):  Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("fromMonth").Select "December"
  MercuryTours\FlightFinder (3,1):  Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("fromDay").Select "29"
  MercuryTours\FlightFinder (4,1):  Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("toPort").Select "San Francisco"
  MercuryTours\FlightFinder (5,1):  Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("toMonth").Select "December"
  MercuryTours\FlightFinder (6,1):  Browser("Find a Flight: Mercury").Page("Find a Flight: Mercury").WebList("toDay").Select "31"
```

The image below shows the Search Results pane for API testing.



```
Total matching results found: 8
  C:\TestUserCode.cs (21,26):    /// <param name="sender">The activity object that raised the ExecuteEvent event.</param>
  C:\TestUserCode.cs (24,57):    public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs args)
  C:\TestUserCode.cs (34,26):    /// <param name="sender">The activity object that raised the AfterExecuteStepEvent event.</param>
  C:\TestUserCode.cs (37,66):    public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseEventArgs args)
  C:\TestUserCode.cs (21,26):    /// <param name="sender">The activity object that raised the ExecuteEvent event.</param>
  C:\TestUserCode.cs (24,57):    public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs args)
  C:\TestUserCode.cs (34,26):    /// <param name="sender">The activity object that raised the AfterExecuteStepEvent event.</param>
  C:\TestUserCode.cs (37,66):    public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseEventArgs args)
```

For details, see " Search Results Pane Overview" on page 463.

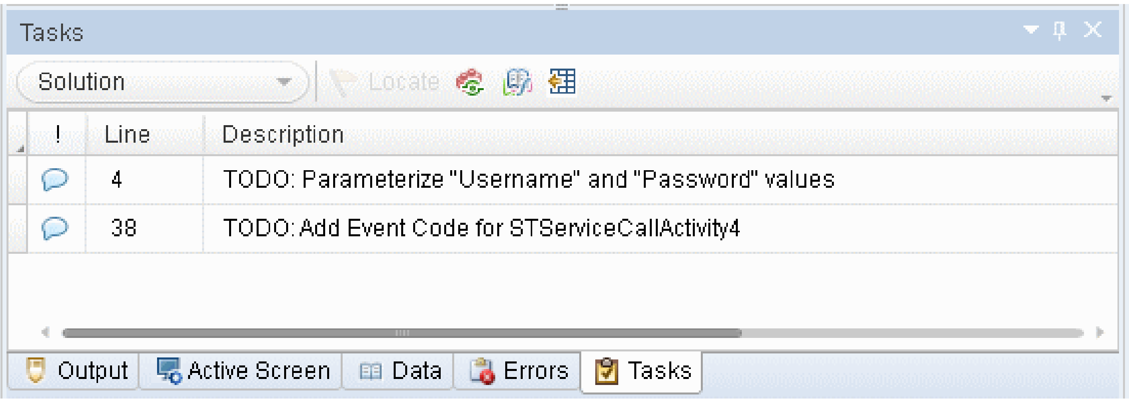
Tasks Pane at a Glance

Relevant for: GUI tests and components and API testing

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

The Tasks pane also enables you to view the TODO comments that exist in an action, an open function library (for example, instructions or notes adjacent to a step), or an open user code file and navigate to the line in the test step or user code file containing the TODO comment.

To show or hide the Tasks pane, select **View > Tasks.**



!	Line	Description
💬	4	TODO: Parameterize "Username" and "Password" values
💬	38	TODO: Add Event Code for STServiceCallActivity4

For details, see "Tasks Pane User Interface" on page 501.

Toolbox Pane at a Glance

Relevant for: GUI tests and components, API testing, and business process testing

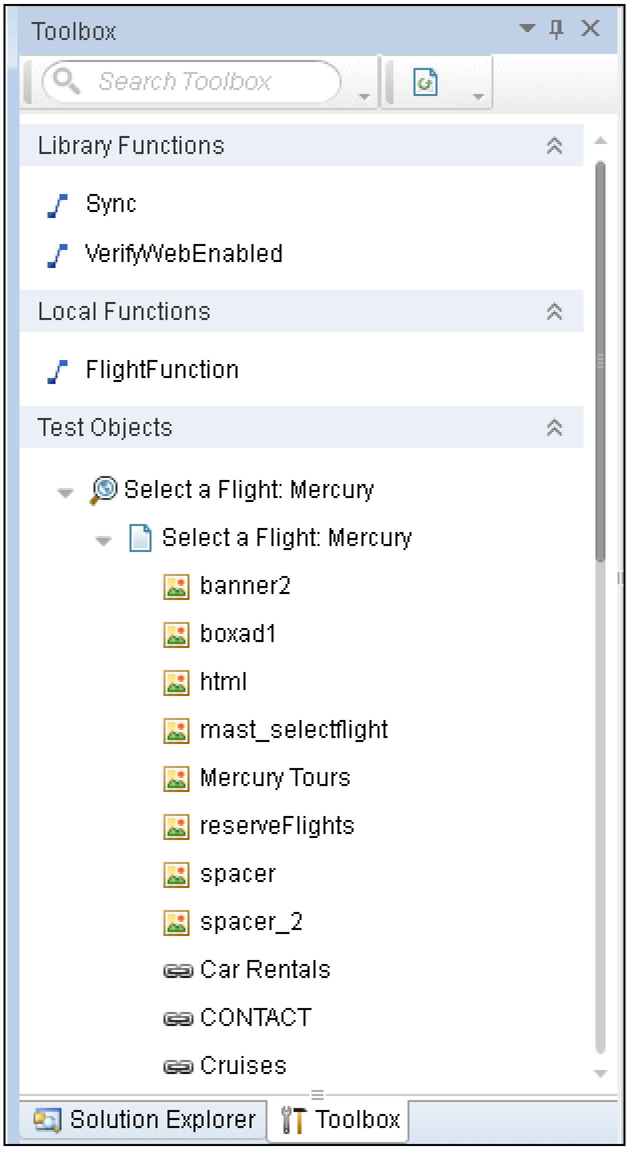
For GUI testing: The Toolbox pane displays all the keywords and functions available to your test or component. It enables you to drag and drop objects or calls to functions into your test or component. When you drag and drop an object into your test or component, UFT inserts a step with the default operation for that object. When you drag and drop a function into your test or component, UFT inserts a call to that function.

For API testing: The Toolbox pane contains all the activities and flow control activities you can use to create an API test. You can drag and drop activities from the Toolbox pane into the canvas to create a test, or you can double-click activities to create a test flow. Custom activities (such as Web services) are also added to the Toolbox pane, under the **Local Activities** node.

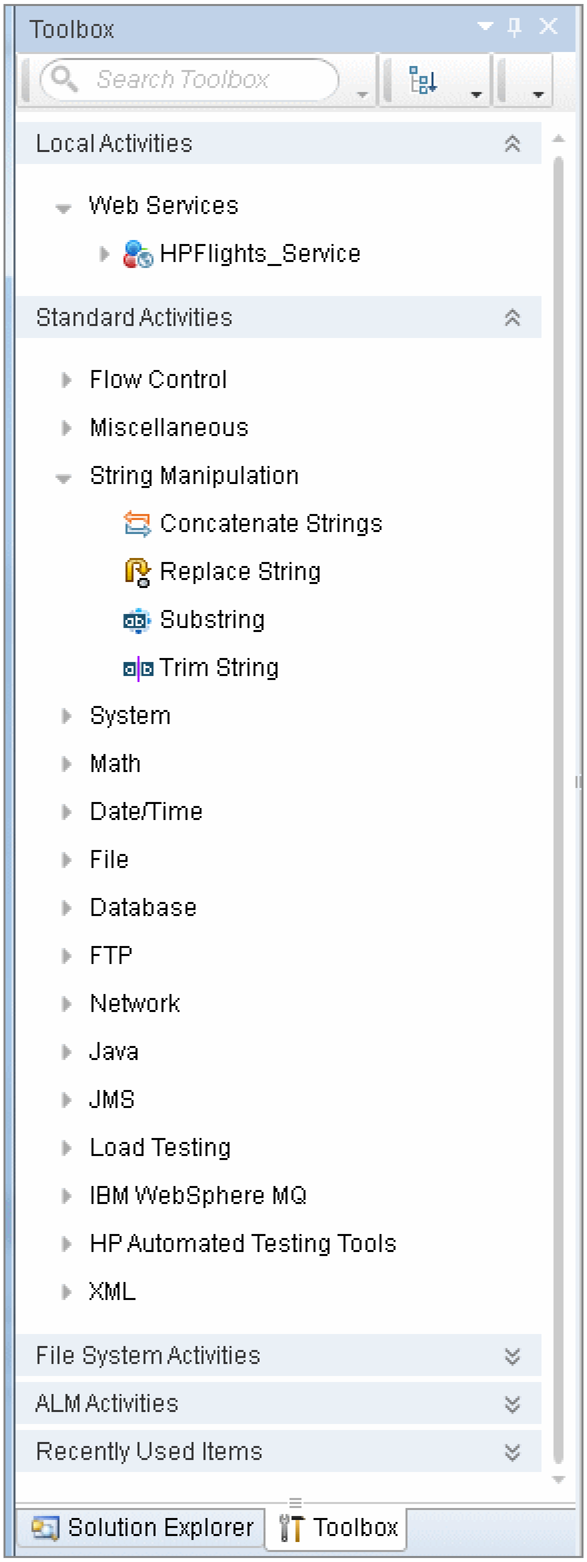
For BPT in UFT: The Toolbox pane displays all of the components and flows available to your business process test or flow. Double-click an item to add it to your test or flow.

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

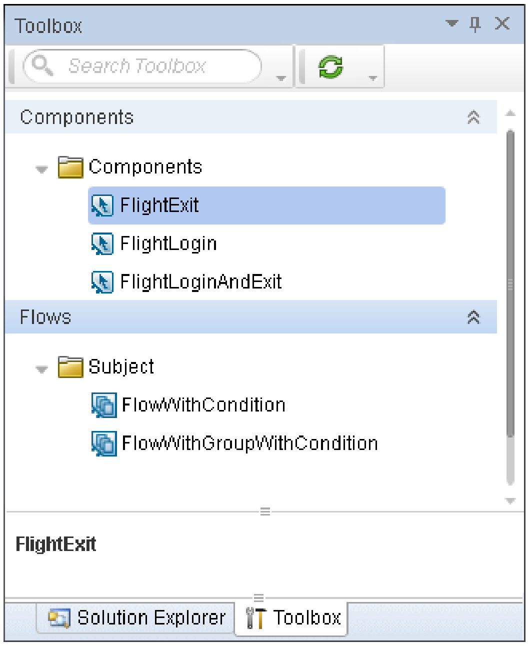
The image below shows the Toolbox pane for GUI testing.



The image below shows the Toolbox pane for API testing



The image below shows the Toolbox pane for BPT:



For details on the **GUI testing** Toolbox pane, see "Toolbox Pane User Interface (GUI Testing)" on page 510.

For details on the **API testing** Toolbox pane, see "Toolbox Pane User Interface (API Testing) " on page 514.

For details on the **BPT** Toolbox pane, see "Toolbox Pane User Interface (BPT in UFT)" on page 518.

Tasks

How to Start UFT

Relevant for: GUI tests and components and API testing

1. From the **Start** menu, select **All Programs > HP Software > HP Unified Functional Testing > HP Unified Functional Testing**, or double-click the **HP Unified Functional Testing** shortcut on your desktop.

   Whenever you start UFT, the Add-in Manager dialog box opens, displaying the currently installed add-ins—unless you specify otherwise. For a user interface description, see "Add-in Manager Dialog Box" on page 118.

   > **Note:**
   >
   > - For details on how to Start UFT from the Windows 8 Start screen, see "Accessing UFT in Windows 8 Operating Systems" on page 75.
   >
   > - If you have not installed a license as part of the installation, you must install a license before opening UFT. For details on the License Wizard, see "Understanding the Unified Functional Testing License Installation Wizard" on page 2184.

2. In the Add-in Manager dialog box, select the add-ins you want to load and click **OK**. The UFT window opens, displaying the "Start Page". For details on the Start Page, see "Start Page" on page 121.

   > **Note:** For details on installing, loading, and working with add-ins, see the *HP Unified Functional Testing Installation Guide* and the *HP Unified Functional Testing Add-ins Guide*.

3. If your installation of UFT integrates with ALM, connect to your ALM project, as described in "How to Work with Tests and Components in ALM" on page 719.

4. Begin creating or editing your tests. For details on how to manage testing documents, see "How to Create and Manage Documents" on page 136.

5. Set your global test options. For details on setting test options in the Options Dialog Box, see "Global Options - Overview" on page 523.

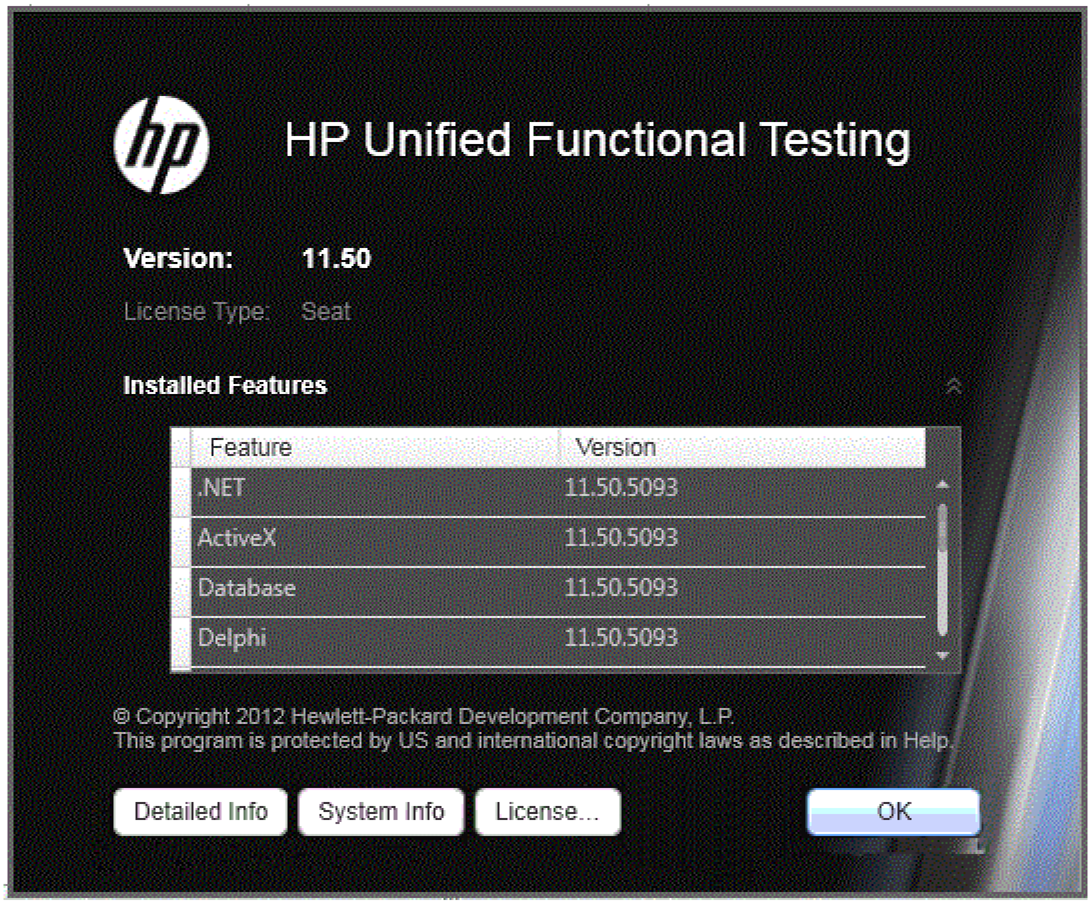
6. **For GUI testing:** Set your individual test or component settings. For details on setting test or component settings, see "Settings Overview" on page 581.

Reference

About Unified Functional Testing Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to view information regarding the features and add-ins, installed on your computer, as well as other basic information about your computer. This information is useful for troubleshooting and when working with HP Software Support.



HP Unified Functional Testing

Version: 11.50

License Type: Seat

Installed Features

Feature	Version
.NET	11.50.5093
ActiveX	11.50.5093
Database	11.50.5093
Delphi	11.50.5093

© Copyright 2012 Hewlett-Packard Development Company, L.P.
This program is protected by US and international copyright laws as described in Help.

To access	Select **Help > About HP Unified Functional Testing**.

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

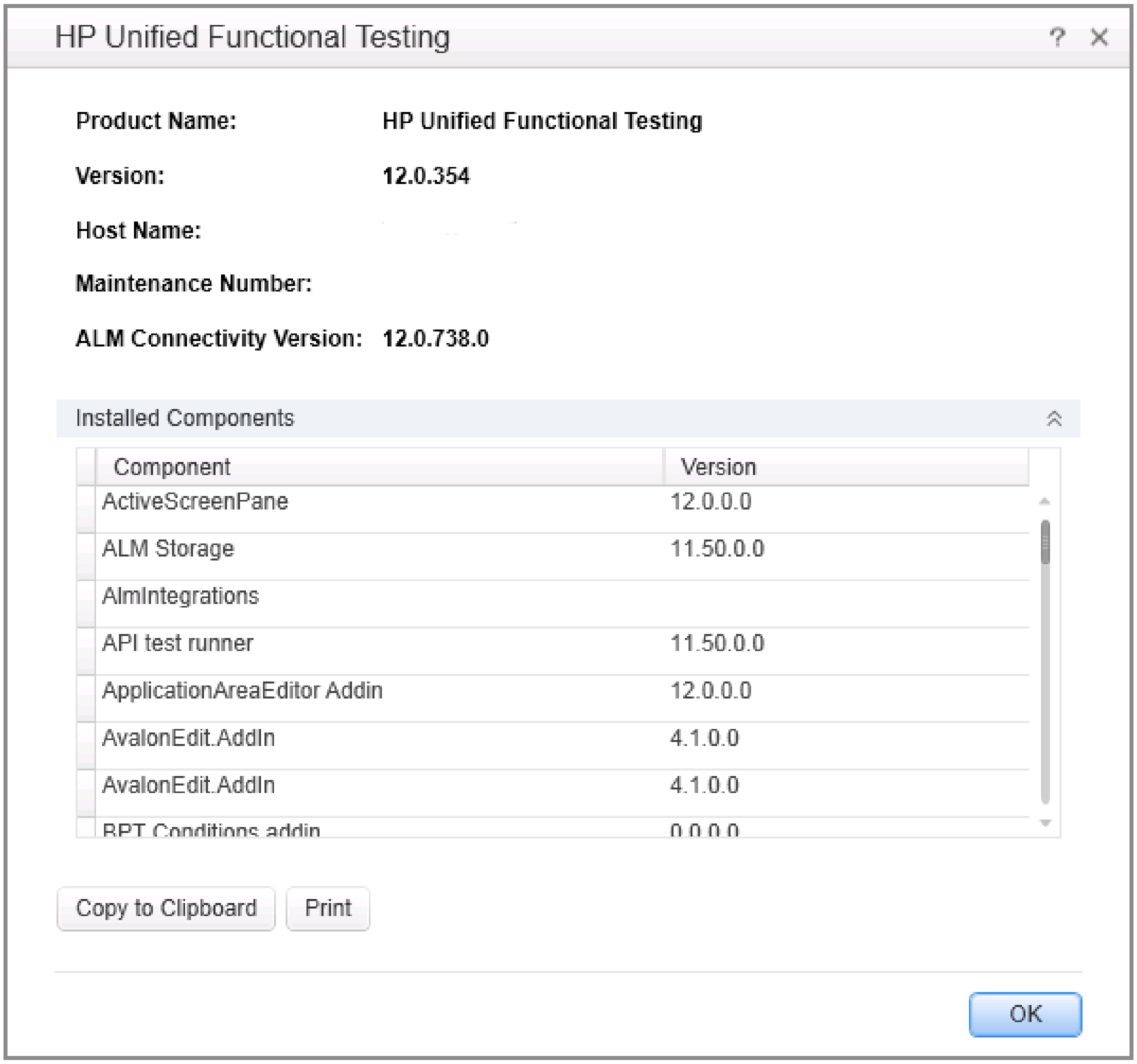
UI Element	Description
Version information	The version of UFT that is installed on your computer and its build number.

UI Element	Description
License Type	Displays the license type installed on your computer: **Seat** or **Concurrent**.
Installed Features	The list of installed add-ins that are installed on your computer. This window displays the feature name and version number for each add-in.
Detailed Info	Opens the "About Unified Functional Testing Detailed Information Window" (described on page 117), which displays all information about the application components installed on your computer.
System Info	Opens the Windows system information window which displays information about the hardware and software components running on your computer.
License	Opens the License Summary dialog box detailing the installed license. Select **Modify License** to change the type of license.

About Unified Functional Testing Detailed Information Window

Relevant for: GUI tests and components and API testing

This window displays detailed information on UFT components installed on your computer.



HP Unified Functional Testing

Product Name: HP Unified Functional Testing

Version: 12.0.354

Host Name:

Maintenance Number:

ALM Connectivity Version: 12.0.738.0

Installed Components

Component	Version
ActiveScreenPane	12.0.0.0
ALM Storage	11.50.0.0
AlmIntegrations	
API test runner	11.50.0.0
ApplicationAreaEditor Addin	12.0.0.0
AvalonEdit.AddIn	4.1.0.0
AvalonEdit.AddIn	4.1.0.0
BPT Conditions addin	0.0.0.0

To access	In the About UFT dialog box, click the **Detailed info** button.
See also	"About Unified Functional Testing Dialog Box" on page 115

User interface elements are described below:

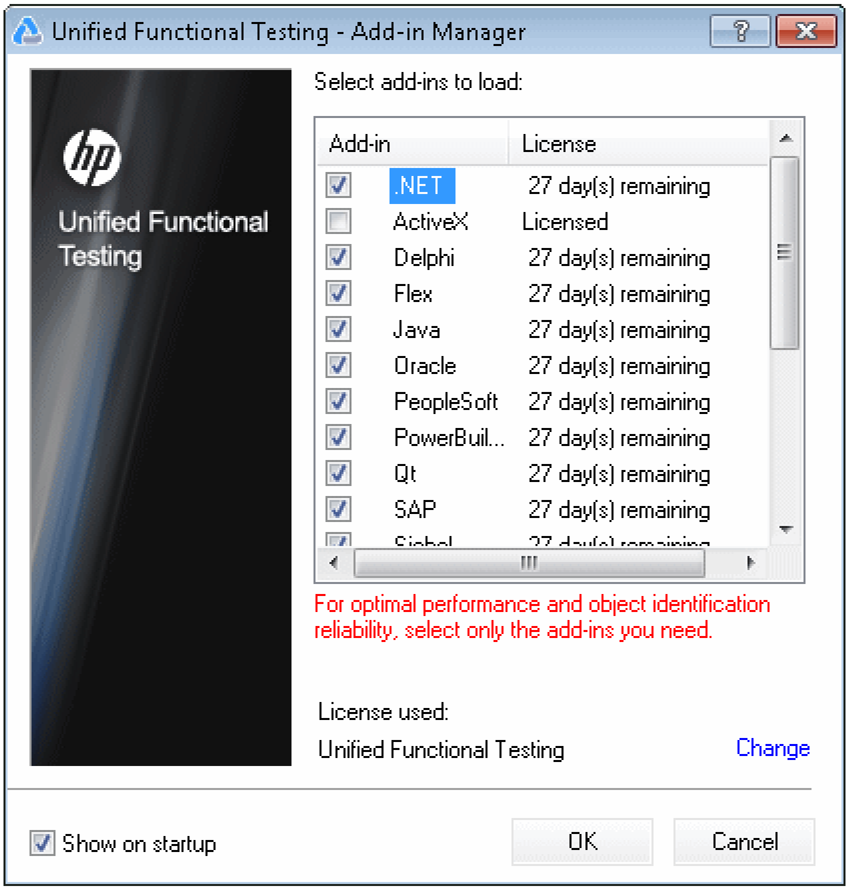
UI Elements	Description
Product Name	The name of the product.

UI Elements	Description
Version	The version number of the product listed in this window.
Host Name	The name of the computer hosting UFT and the operating system.
ALM Connectivity Version	The version number of the ALM Connectivity Add-in.
Maintenance Number	The maintenance number for your software.
Installed Components	The list of UFT components installed on your computer, including version number.

Add-in Manager Dialog Box

Relevant for: GUI tests and components and business process tests and flows

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.



Unified Functional Testing - Add-in Manager

Select add-ins to load:

	Add-in	License
☑	.NET	27 day(s) remaining
□	ActiveX	Licensed
☑	Delphi	27 day(s) remaining
☑	Flex	27 day(s) remaining
☑	Java	27 day(s) remaining
☑	Oracle	27 day(s) remaining
☑	PeopleSoft	27 day(s) remaining
☑	PowerBuil...	27 day(s) remaining
☑	Qt	27 day(s) remaining
☑	SAP	27 day(s) remaining
☑	Siebel	27 day(s) remaining

For optimal performance and object identification reliability, select only the add-ins you need.

License used:
Unified Functional Testing

☑ Show on startup

To access	By default, this dialog box opens when you start UFT. To display the Add-in Manager if it does not open when you start UFT, select **Tools > Options > General** tab **> Startup options** node and then select **Display Add-in Manager on startup**.
Important information	- 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.
Relevant tasks	"How to Start UFT" on page 114
See also	"About Unified Functional Testing Detailed Information Window" on page 117

User interface elements are described below:

UI Element	Description
Add-in	The names of the installed add-ins. 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 HP Unified Functional Testing Extensibility Documentation program group (**Start > All Programs > HP Software > HP Unified Functional Testing > Extensibility > Documentation**). **Note:** - 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, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

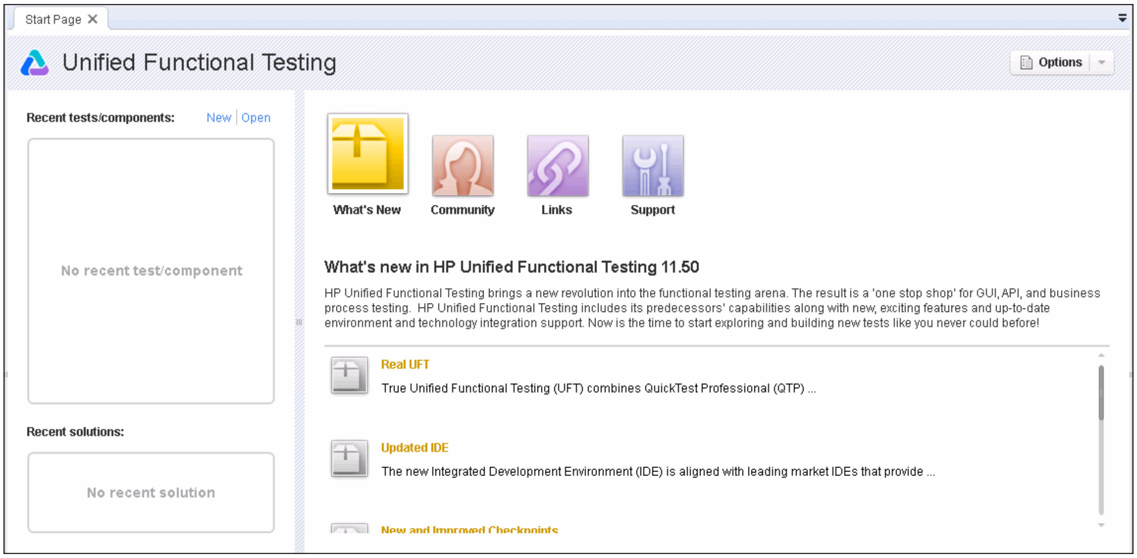
UI Element	Description
License	The license used by the add-in, if any, and the time remaining until a time-limited license expires: • **Licensed.** Applies to the add-ins that are provided with UFT. Add-ins use the same license as UFT. Therefore, if UFT uses a **Permanent** license, the add-ins use the 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.) For details, see the *HP Unified Functional Testing Installation Guide***.**
License used (concurrent licenses only)	The license used by UFT: • **Unified Functional Testing**.Enables you to use all UFT features, including the use of GUI testing and API testing features. • **QuickTest Professional**. Enables you to open only GUI tests and components. When using this license, API testing-related options are not available. • **Service Test**. Enables you to open only API tests and components. When using this license, GUI testing-related options are not available. • **QuickTest and Service Test.** Enables you to use UFT with a QuickTest 11.00 or earlier or a Service Test 11.20 or earlier license. When using this type of license, business process testing features and GUI/API integration features (like calls to a GUI test from an API test and vice versa are not available). Click **Change** to modify the type of license used with a session of UFT. **Note:** • You can only choose the type of license when working with a concurrent license. This menu is hidden when using a seat license. • To work with BPT in UFT, select Unified Functional Testing.

UI Element	Description
Show on startup	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. **Note: (for concurrent license users)** If this check box was cleared in the previous session, and the license type selected from the concurrent license server in that session is not currently available, UFT tries to load an available license that matches the selected add-ins. To display the Add-in Manager again: Select **Tools > Options > General** tab > **Startup options** node and select **Display Add-in Manager on startup**.

Start Page

Relevant for: GUI tests and components and API testing

This page welcomes you to UFT and describes the new features in this release, including links to more information about these features.



What's new in HP Unified Functional Testing 11.50

HP Unified Functional Testing brings a new revolution into the functional testing arena. The result is a 'one stop shop' for GUI, API, and business process testing. HP Unified Functional Testing includes its predecessors' capabilities along with new, exciting features and up-to-date environment and technology integration support. Now is the time to start exploring and building new tests like you never could before!

Real UFT

True Unified Functional Testing (UFT) combines QuickTest Professional (QTP) ...

Updated IDE

The new Integrated Development Environment (IDE) is aligned with leading market IDEs that provide ...

New and Improved Checkpoints

To access	1. Start UFT, as described in "How to Start UFT" on page 114. 2. If the Start Page window is not displayed, select **View > Start Page.**

User interface elements are described below:

UI Elements	Description
Recent Tests/Components **Recent Solution**	Enables you to create a test or component or a solution by clicking the **New** or **Open** buttons after selecting a test, component or solution name.
Options	Options for viewing the start page. You can select **Display the Start Page on Startup** to display the Start Page immediately upon the launch of UFT. You can select **Close Start Page after test loads** if you want to close the start page after loading a test.
What's New area	Click the **What's New** button to display information on the list of the newest features, enhancements, and supported environments in the latest version of UFT.
Community Area	Click the **Community** button to open news about UFT, links to UFT in the HP Software site, links to the GUI Testing Forum, the API Testing forum, and the BPT Forum, or the UFT blog.
Links Area	Click the **Links** button to open links to the UFT Help, the *HP UFT Object Model Reference for GUI Testing*, and the tutorials for GUI Testing and API Testing.
Support Area	Click the **Support** button to open links to UFT support, the Knowledge Base, and Troubleshooting areas of the HP Software Support Web site.

UFT Program Folder Structure

Relevant for: GUI tests and components and API testing

After the UFT setup process is complete, the following items are added to your UFT program folder (**Start > All Programs > HP Software > HP Unified Functional Testing**):

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

Option	Description
HP Unified Functional Testing Code Samples Plus	Opens the Unified Functional Testing Code Samples Plus Help, which provides sample function libraries, code, and SDK samples with accompanying explanations. **Note:** These samples are relevant for GUI testing only.
HP UFT Help	Opens the UFT Help, which displays links to commonly used topics and movies that describe how to use UFT, as well as additional links to HPSoftware Websites. The UFT Help provides access to all guides available for UFT, such as Getting Started guides, Helps, reference files, and links to printer-friendly (PDF format) documentation. It contains various navigation options to help you find the information you need. The **Welcome to the UFT Help** section describes how to navigate, use, and get updates for the UFT Help.
Unified Functional Testing Automation Reference	Opens the Unified Functional Testing Automation Object Model Reference for GUI Testing. The object model assists you in automating GUI test management, by providing objects, methods and properties that enable you to control UFT features and configurations. The Object Model Reference provides syntax, descriptive information, and examples for the objects, methods, and properties. It also contains a detailed overview to help you get started writing GUI test scripts.
UFT Tutorials	Opens the GUI testing and API testing tutorials, which teach you basic skills and shows you how to start testing your applications.

- **Sample Applications.** Contains the following links to sample applications that you can use to practice testing with UFT:

Option	Description
Flight API	Opens a GUI-less flight service application, used in conjunction with an API test. **Note:** You must have administrator privileges to use this application.
Flight GUI	Opens a sample flight reservation Windows application. To access the application, enter any username and the password **mercury**.

Option	Description
Mercury Tours Web site	Opens a sample flight reservation Web application. This Web application is used as a basis for the UFTGUI Testing tutorial. For details, see the *HP Unified Functional Testing Tutorial for GUI Testing*.

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

> **Note:** The available tools depend on the installed UFT add-ins.

Option	Description
Activity Wizard	Opens the API Testing "Activity Wizard" (described on page 2049), which enables you to create customAPI activities that will be visible in the Toolbox pane.
Additional Installation Requirements	Opens the "Additional Installation Requirements Utility" (described on page 2181), which displays any prerequisite software that you must install or configure to work with UFT. For details, see the
Commuter License Tool	Opens the Commuter Licensing utility, to check in or check out concurrent licenses when working at a remote location.
HP Micro Player	Opens the HP Micro Player, which enables you to view captured movies of a run session without opening UFT. For details, click the **Help** button in the HP Micro Player window.
Java Add-in JRE Support Tool (GUI testing only)	Opens the Java Add-in JRE Support Tool, which adjusts the JVM Runtime Parameters of your JRE to enable the Java Add-in to recognize Java applets and the Java objects within them. This tool is necessary only for certain operating systems, browsers, and JRE versions. For details, see the Java section of the *HP Unified Functional Testing Add-ins Guide*. This tool is only available when the Java add-in is installed with UFT.
License Validation Utility	Opens the License Validation utility, which enables you to retrieve and validate license information. For details, click the **Help** button in the License Validation Utility window.
Password Encoder (GUI testing only)	Opens the Password Encoder tool, which enables you to encode passwords. You can use the resulting strings as method arguments or Data pane parameter values. For details, see "Password Encoder Tool" on page 965.

Option	Description
Register New Browser Control (GUI testing only)	Opens the Register Browser Control Utility, which enables you to register your browser control application so that UFT recognizes your Web objects when recording or running GUI tests. For details, see the section on registering browser controls in the *HP Unified Functional Testing Add-ins Guide*.
Remote Agent	Activates the UFT Remote Agent, which enables you to configure how UFT behaves when a GUI test or component is run by a remote application such as ALM. For details, see "Remote Agent Settings Dialog Box" on page 745.
Silent Test Runner (GUI testing only)	Opens the "Silent Test Runner" (described on page 2239). This enables you to run a test in the manner in which it is run from LoadRunner or Business Availability Center.
soapUI to API Test Converter (API Testing only)	Converts soapUI tests to a UFT API Test.
Stingray Support Configuration Wizard (GUI testing only)	Opens the Stingray Support Configuration Wizard which enables UFT to recognize Stingray objects in your application. This tool is only available when the Stingray add-in is installed with UFT.
Test Batch Runner	Opens the Test Batch Runner application, which enables you to set up UFT to run several tests in succession. For details, see "How to Create and Run a Test Batch" on page 646.

- **HP Unified Functional Testing Product Availability Matrix.** Provides a complete list of all environments, programs, and versions that are supported for UFT and its add-ins.

- **HP Unified Functional Testing**. Opens the UFT application.

- **Readme.** Opens the *HP Unified Functional Testing Readme*, which provides the latest news and information on UFT and the UFT add-ins.

Note:

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

- For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

Chapter 3: UFT File Operations

Relevant for: GUI tests and components and API testing

This chapter includes:

Concepts

Testing Documents Overview

Relevant for: GUI tests and components and API testing

A testing **document** is any file that enables you to test your application. You create, maintain, and edit steps, parameters, functions, and other details in the document to enable the testing of your application.

Testing documents can include:

- **GUI testing documents:**
  - GUI tests
  - GUI actions
  - Function libraries
- **API testing documents:**
  - API tests
  - API actions
  - C# files containing event handler code (TestUserCode.cs files)
  - C# files containing user-generated code
- **BPT documents:**
  - Business process tests
  - Business process flows
  - Business components, including API, keyword GUI, and scripted GUI components
  - Application areas

You work with testing documents in the "Document Pane" (described in "Document Pane" on page 302). In UFT, you can perform standard file operations with testing documents, including creating, opening, saving, and printing multiple types of documents. You can also save a test with its resources, save your test files in a **.zip** format, or import a test document that has been stored as a **.zip** file.

The document pane supports the following types of views and functionality:

- **Canvas.** Graphically displays and enables you to edit the flow of your test, action, or component. For details, see "The Canvas" on page 209.
- **Keyword View.** Enables you to create and view the steps of your action or component in a keyword-driven, modular, table format. For details, see "Keyword View User Interface" on page 942.
- **Editor.** Provides text and code editing features that facilitate the design of your text, script, and code documents. For details, see "Editing Text and Code Documents" on page 305.
- **Application area.** Displays and enables you to edit the application area settings and resource associations. For details, see "Application Area User Interface" on page 2101.
- **BPT in UFT.** Displays and enables you to create and edit business process tests and flows that are stored in ALM. For details, see "Business Process Testing in UFT User Interface" on page 2074.

Relative Paths in UFT for GUI Testing

Relevant for: GUI tests and components

UFT enables you to define the path to a GUI testing resource stored in the file system or in ALM as a relative or an absolute path.

If you specify a relative path, then during a run session, UFT searches for the resource file in the folder for the current test or component, 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 "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544.

Note:

- A path that starts with \.. is considered to be a full path, with the backslash representing the root folder of the current drive.
- If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, specify an absolute Quality Center or ALM path. For details, see "Relative Paths and ALM" on page 758.

Example of When to Use Relative Paths

If your test's resources are stored in the file system and you want other users or HP products to be able to run this test on other computers, you can set the file path to the resource as a relative path. Any users who want to run this test should then specify the drive letter and folder in which UFT should search for the relative path in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node).

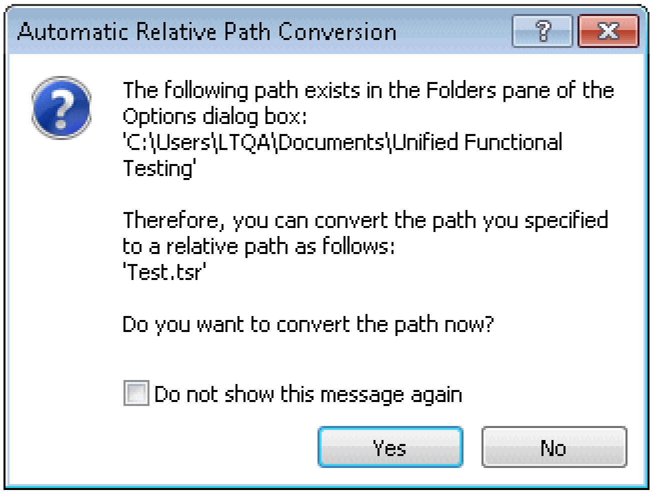
Entering a Path to a Resource

When you specify a path to a function library, shared object repository, recovery scenario, data table file, or environment variable file, UFT checks if the path, or the initial part of the path, exists in the Folders pane of the Options dialog box.

UFT then enables you to specify whether to convert the path to a relative path or store an absolute path. One of the following message boxes opens, depending on whether the path you specified, or a part of the path, exists in the Folders pane.

Path Exists in the Folders Pane

If the resource path you specify matches an existing search path in the Folders pane, you are prompted whether to define the path using only the relative part of the path.



Automatic Relative Path Conversion

The following path exists in the Folders pane of the Options dialog box:
'C:\Users\LTQA\Documents\Unified Functional Testing'

Therefore, you can convert the path you specified to a relative path as follows:
'Test.tsr'

Do you want to convert the path now?

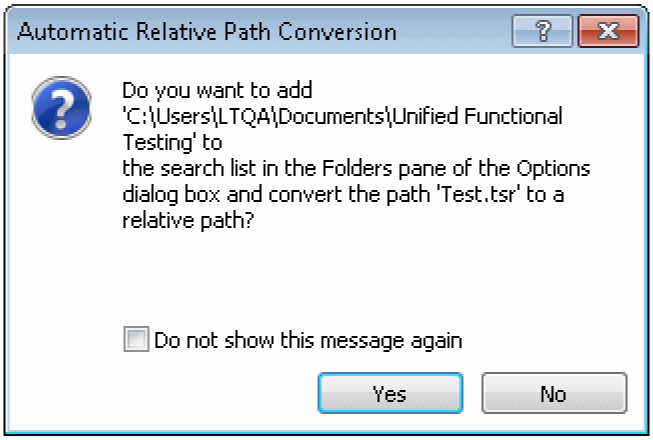
□ Do not show this message again

- Clicking **Yes** truncates the path to a relative path.
- Clicking **No** defines the path to the resource as an absolute path.

In cases where a part of the path you enter matches more than one path in the Folders pane, the closest match is applied. For example, if both C:\Current_Version and C:\Current_Version\Libraries are defined in the search path list, the latter is applied.

Path Does Not Exist in the Folders Pane

If the resource path you specify does not match an existing search path in the Folders pane, you are prompted whether to add the resource's location path to the Folders pane and define the path relatively.



Do you want to add 'C:\Users\LTQA\Documents\Unified Functional Testing' to
the search list in the Folders pane of the Options dialog box and convert the path 'Test.tsr' to a relative path?

□ Do not show this message again

- Clicking **Yes** adds the resource's location path to the Folders pane and truncates the path to a relative path.

- Clicking **No** defines the path to the resource as an absolute path.

> **Note:**
>
> - You can choose not to show one or both of these message boxes when you enter a path to a resource by selecting the **Do not show this message again** check box. To show these message boxes again, select the **Remind me to use relative paths when specifying a path to a resource** check box in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node). This check box is selected by default when you first start UFT.
>
> - If you are connected to Quality Center 10.00 or ALM, these message boxes are displayed only if you select a path in the file system.

Understanding Absolute and Relative Paths

Relevant for: GUI tests and components

You can reference UFT GUI testing resources, such as shared object repositories, function libraries, recovery scenarios or environments, using absolute or relative paths.

- An **absolute** path describes the full path to a specific file starting from a fixed location such as the root folder, or the drive on which the file is located, and contains all the other subfolders in the path. An absolute path always points to the specified file, regardless of the current folder.

If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, you must specify an absolute Quality Center or ALM path to enable your tests to access your resource files.

- A **relative** path describes the path to a specific file starting from a given folder, and is generally only a portion of the absolute path. A relative path therefore specifies the location of the file relative to the given location in the file system.

  Using relative paths means that the paths remain valid when files or folders containing files are moved or copied to other locations or computers, provided that they are moved within the same folder structure. If you are not working with the Resources and Dependencies model with Quality Center 10.00 or ALM, we recommend that you use relative paths when saving resources in UFT.

Example

Consider a UFT GUI testing resource file named FunctionLibrary1.qfl located in C:\Current_Version\Libraries. The absolute path to the file is C:\Current_Version\Libraries\FunctionLibrary1.qfl. The relative path to the file from within the folder named Libraries is specified using only the name of the file, FunctionLibrary1.qfl. Alternatively, the relative path to the file from within another folder, such as C:\Current_Version\Libraries\MyFiles, would be Libraries\FunctionLibrary1.qfl.

Using a relative path, you could copy the FunctionLibrary1.qfl file from C:\Current_Version\Libraries to an updated version in C:\New_Version\Libraries, and the path used by UFT would remain valid.

Portable Copies of Tests

Relevant for: GUI tests and API tests

When you work with a GUI test, the tests and their resource files are often stored on a network drive or in ALM, as this enables the reuse of actions and other resources, and helps ease test management. Therefore, you may need to open or run a test when you do not have access to a network drive or ALM. For example, you may need to create a portable copy of a test for use when traveling to other sites.

For API tests, some of the activities and reference files used with a test are stored locally in the same location as the test. Therefore, when you move the test to another computer, UFT cannot locate the resources, and therefore not run the test.

For GUI tests

You can save a standalone copy of your GUI test and its resource files to a local drive or to another storage device using the **File > Save (Other) > Save with Resources** command.

When you save a test in this manner, UFT creates a copy of the following and saves the files in the location you specify:

- **Source test.** UFT saves a copy of this test in the location you specify.
- **Resource files.** UFT saves a copy of all resource files associated with the source test, such as function libraries and shared object repositories. UFT stores these files in sub-folders of the copied test.
- **Called actions.** UFT saves a copy of any external actions called by the source test. For example, if Test A calls actions that are stored in Test B, UFT creates a local copy of the actions stored in Test B and stores them in a sub-folder of Test A. The sub-folder has the same name as the test from which the called actions were copied. In this example, the sub-folder is named Test_B. UFT also creates a copy of any resources associated directly with these actions, such as its local shared object repositories and action sheets in the data table. UFT does not, however, save the resource files associated with Test B, so you must ensure that these resources are associated with the source test, Test A.

This enables you to modify or run the test without access to a network drive or ALM.

For API tests

You can save a standalone copy of your API test and its resources to a local drive or to another storage device by saving the activity as a local activity. For details, see "Save a portable copy of an API test" on page 139.

> **Tip:** If you use UFT with a concurrent license but do not have access to the concurrent license server (for example, during a business trip), you can install a commuter license. For more details, see the *HP Unified Functional Testing Installation Guide*.

Opening and Saving Tests with Locked Resources

Relevant for: GUI tests and API tests

Resource files can be locked by UFT to protect the information in the file from being overwritten.

Resource files are locked in the following scenarios:

- The file is being used by another UFT user.
- The file is checked into an ALM version control database or any other version control software.
- The file is marked as read-only.

> **Note:** External files that are associated with your test but cannot be edited within UFT (such as environment variable files) are not affected by this locking mechanism.

UFT notifies you if a resource file that is associated with your test is locked when:

- Opening a test whose resource file is locked. For details, see "Open Test with Locked Resources - Message Box" on page 163.
- Saving a test whose external resource files were not opened in read-write mode (because they were locked when you opened the test) and modified while editing the test. For details, see "Save Test with Locked Resources - Message Box" on page 168.

> **Tip:** To create a writable copy of a test with locked resources, use the Save As option from the File menu and save the test under a new name.

Tasks

How to Create and Manage Documents

Relevant for: GUI tests and components, API testing, and business process testing

The task describes how to manage testing documents and includes the following steps:

- "Create a new standalone document" below
- "Create a new document within an existing solution" on the next page
- "Open an existing document" on the next page
- "Add an existing document to your solution" on page 138
- "Edit an open document" on page 138
- "Save the current document" on page 138
- "Save all open documents" on page 138
- "Save a portable copy of an API test" on page 139
- "Save a portable copy of a GUI test" on page 139
- "Close the current document" on page 139
- "Close all documents" on page 140
- "Switch between open documents" on page 140
- "Zip and unzip a test (GUI tests only)" on page 141
- "Delete a document" on page 141

Create a new standalone document

1. (Optional) If you are working with ALM, connect to the relevant ALM project. For details, see "ALM Connection Dialog Box" on page 737.

2. Do one of the following:

   - Select **File > New** and select the type of document to create.

   - Click the **New** button down arrow and select the type of document to create from the drop-down list.

A dialog box opens:

- If you are creating a test or component, see "New <Document> Dialog Box" on page 152.

  This dialog box enables you to specify the type of test or component you want to create, the location in which the new item is stored, and, optionally, the solution in which it is included.

- If you creating a function library, application area, or solution, see "Open/New <Document>/<Resource> Dialog Box" on page 156.

  This dialog box enables you to specify the location in which the new item is stored.

- If you are creating a GUI test from an XML file containing exported Sprinter steps and test objects see "Sprinter Automated Test Data File Dialog Box" on page 150.

  This dialog box enables you to specify the location of the XML file, the name of the test to create, and the location in which to store it.

After you finish entering the information for your new document in the dialog box, a new document is automatically saved and displayed as a tab in the document pane.

Create a new document within an existing solution

Do one of the following:

- Select **File > Add** or click the **Add** down arrow to add a new document to your solution.
- In the "Solution Explorer Pane" (described on page 478) right-click the <**solution name**> node, select **Add > Add New** and select the type of document to create.

A dialog box opens. For details, see: "Add Test/Component to Solution Dialog Box" on page 146, or, if you are creating a GUI test from an XML file containing exported Sprinter steps and test objects see "Sprinter Automated Test Data File Dialog Box" on page 150.

After you finish entering the information for your new document in the dialog box, a new document is automatically saved and displayed as a tab in the document pane.

Open an existing document

1. If your document is stored in ALM, connect to the relevant ALM project. For details, see "ALM Connection Dialog Box" on page 737.
2. Do one of the following:

   - Select **File > Open**, click the **Open** down arrow, or select the list of **Recent** files in the **File** menu to open an existing document. In the "Open/New <Document>/<Resource> Dialog Box" (described on page 156), browse to and select your document.
   - If your document is part of a solution, in the "Solution Explorer Pane" (described on page 478)

double-click the selected node for your document.

> **Note:** Within a solution, different document types are located in different locations. For details, see "Open an existing document from the current solution" on page 475.

If your GUI test has locked resources, see "Opening and Saving Tests with Locked Resources" on page 134 and "Open Test with Locked Resources - Message Box" on page 163.

Add an existing document to your solution

Do one of the following:

- In the "Solution Explorer Pane" on page 470 (described on page 478), right-click the <**solution name**> node and select **Add > Add Existing <Document>**, and select the type of document to add.
- Select **File > Add Existing** or the **Add** down arrow to add a document to your solution.

A dialog box opens. For details, see "Add <Existing Document> to Solution Dialog Box" on page 145.

Edit an open document

The features available for viewing and editing documents depend on the document type. For details, see "Document Pane Overview" on page 303.

Save the current document

Do one of the following:

- Make sure that the document you want to save is the active tab (you can select the document's tab to bring it into focus), and click the **Save** button .
- Right-click the tab of the document you want to save and select **Save <document name>** or **Save <document name> As**.

UFT saves the document with your changes. If you are using **Save As**, the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164) opens.

> **Note:** When you edit a document, an asterisk (*) is displayed in the document tab (for standalone documents like application areas, function libraries, or user code files) or the canvas tab (if the document is part of a test) until the document is saved.

Save all open documents

Select **File > Save All** or click the **Save All** button .

> **Note:** When you edit a document, an asterisk (*) is displayed in the document tab (for

standalone documents like application areas, function libraries, or user code files) or the canvas tab (if the document is part of a test) until the document is saved.

Save a portable copy of an API test

1. Prerequisites:

   - If your test was last modified using Service Test 11.10 or later, or using UFT, make sure that the test and its resources are upgraded to UFT. Open the test in UFT and save it (**Save** or **Save As**). If the test contains calls to external actions (actions stored in other tests), open and save those tests too.

     Note: Tests last modified in Service Test 11.00 or earlier cannot be opened in UFT.

   - Make sure that your resource file (such as WSDL or REST service) has at least one step in the current test.

2. In the Toolbox pane, right-click on the service name and select **Move to > File System Activities**. The service moves to the File System Activities section of the Toolbox pane.

3. Select **File > Save (Other) > Save with Resources**. The "Save <Resource>/Save <Document> As Dialog Box" (described on page 164) opens.

Save a portable copy of a GUI test

1. If your test was last modified using QuickTest 11.00 or earlier, make sure that the test and its resources are upgraded to UFT. Open the test in UFT and save it (**Save** or **Save As**). If the test contains calls to external actions (actions stored in other tests), open and save those tests too.

2. Select **File > Save (Other) > Save with Resources**. The "Save Test with Resources Dialog Box" (described on page 169) opens

3. Select a name and location for the test and click **Save**.

For details about portable tests, see "Portable Copies of Tests" on page 133.

Close the current document

Do one of the following:

- Make sure that the document you want to close is the active document and select **File > Close**.

- To close all documents included in the current solution and the solution, select **File > Close solution**.

- Right-click a document tab and select **Close.**

Close all documents

Do one of the following:

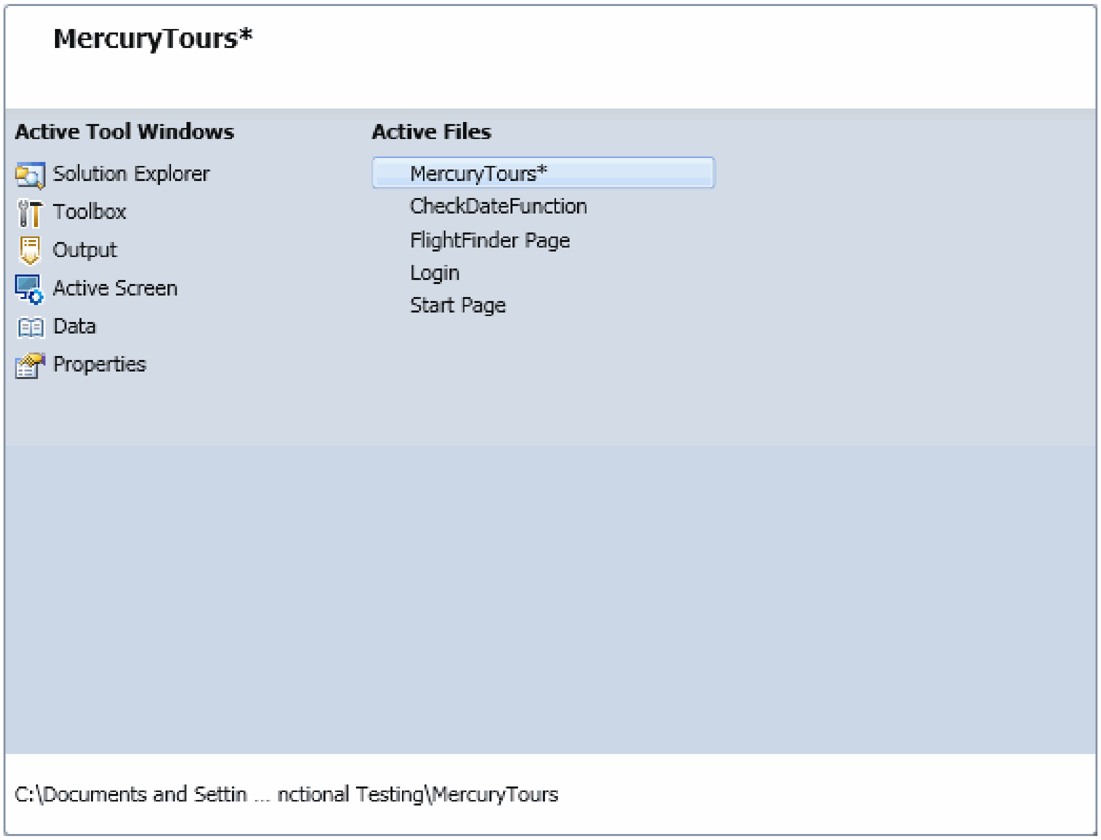
- To close all open documents, right-click any document's tab and select **Close All**.
- To close all open documents except the active document, right-click on the document tab and select **Close All but This**.

Switch between open documents

- To navigate to an open document, select the relevant tab.

> **Note:** If some tabs are not visible due to a lack of space, click the document display arrow ▼ in the upper-right hand corner to display a list of all open documents. Select the document you want to view.

- To select your document from a menu of open documents, press and hold CTRL+TAB on your keyboard. A window opens showing a list of all open documents and panes.



Zip and unzip a test (GUI tests only)

- To **zip** a test, select **File > Export Test** and use the "Export to Zip File Dialog Box" (described on page 151).

  This enables you to save configuration, run-time, setup data, and Active Screen files together, conserving space and making it easier to move a test.

- To **unzip** a test, select **File > Import Test** and use the "Import from Zip File Dialog Box" (described on page 152).

Delete a document

- If the document is stored in the file system, select **File > Open**, **File > New**, or **File > Save as** to open a file operation dialog box. Browse to the document, right-click and select **Delete**.

  > **Caution:**
  >
  > - **For GUI tests:** Do not delete actions in this way. Delete them from within the test that contains them.
  > - Before you delete a document, make sure that it is not used by or associated to any other documents.

- If the document is stored in ALM, you delete it from within ALM, regardless of whether it was created in UFT or in ALM. For details, see *HP Business Process Testing User Guide*.

How to Define API Test Properties or User/System Variables

Relevant for: API testing only

The following steps describe how to define test properties, user variables, and operating system variables. These settings are optional.

- "Define test parameters" below
- "Define user variables" on the next page
- "Set user variable values " on the next page
- "Define user variable profiles " on page 143
- "Set OS variable values for the test - optional" on page 143

Define test parameters

This step applies if you want to define custom parameters that can be used by all steps in the test, with the ability to assign data from a data source.

1. Click in a blank area of the canvas. In the Properties pane, open the **Test Input Parameters** view .

2. Click the **Add** button to define a new input or output parameter.

3. In the "Add Input/Output Property/Parameter Dialog Box (API Testing)" (described on page 1775), specify a parameter name and data type. All types that were defined in any reference file, are available.

4. Click in the **Default Value** column and specify a value.

Define user variables

You can create user variables and set their values. You can define multiple profiles for the variable values. You select a profile to be active before the test run. For details, see "Set user variable values " below.

1. Click in a blank area of the canvas. In the Properties pane, open the **Test Variables** tab .

2. Click the **Add User Variable** button to define a new user variable.

Set user variable values

You can set values for variables in one of the following ways:

- In the Properties pane's **Test Variables** view, click in the **Profile** column and manually enter values.

- Open the Toolbox pane and drag the **Set Test Variable** activity from the **Miscellaneous** category into the **Test Flow** (or loop). In the Properties pane, define the variable key and value. To obtain values, click the Link to a Data Source button in the row. In the "Select Link Source Dialog Box (API Testing)" (described on page 1840), select the **Test Variables** option. Select a name and value for the **Variable key** and **Variable value**.

- Open the Properties pane's **Events** view for the step or define a Custom Code activity. Edit the event handler code and assign a value using the TestProfile object. The following example sets the value of the Region user variable to NE.

  ```
  activity.Context.TestProfile.SetVariableValue("Region", "NE");
  ```

For details, see "Writing Event Handlers for API Test Steps" on page 1935.

Repeat this step for each variable for which you want to set values.

Define user variable profiles

This step applies only if you created user variables as described in the above steps.

1. Define one or more user variables as described above.

2. Click the **Add New Profile** button .

3. In the "New Test Profile Dialog Box" (described on page 172), specify a name and indicate the profile (if any) from which to copy the properties. If you do not copy the properties from an existing profile, UFT copies the user variables from the active profile without its values.

4. To rename or delete a profile, click the **Manage Profiles** button . Click **Remove** or **Rename**.

5. Click the **Compare** button to display the profiles side-by-side.

6. Only the active profile is accessed during the test run. To make a profile active, select it from the **Active Profile** list.

   For user interface details, see the "Test Variables Tab (Properties Pane - API Testing)" on page 432.

Set OS variable values for the test - optional

This step lets you set global operating system variables that will apply to all steps in the current test run.

1. From the Toolbox pane and drag the **Set OS Environment Variable** activity from the **System** category into the Test Flow or another custom loop.

2. In the Properties pane's Input/Checkpoints tab, define the variable key and value or click the **Link to Source** button to specify a data source.

   > **Tip:** To view a list of the test variables, click in a blank area of the canvas. In the Properties pane, open the **Test Variables** tab . The System variables are listed in the lower pane.

How to Upgrade API Tests Using the Batch Upgrader

Relevant for: API testing only

UFT provides an upgrade tool for API tests, STBatchUpgrader.exe, located in the <Unified Functional Testing installation>/bin folder. This tool lets you run a batch file to upgrade tests last modified in version 11.10 or 11.20, making them compatible for UFT.

> **Note:** The Batch Upgrader tool may not successfully upgrade tests stored in ALM11.00 when working on Windows 2003.

If you do not upgrade your tests with the batch upgrade tool, when you open a test last modified in version 11.10 or 11.20, it prompts you to upgrade the test.

Tests last modified in Service Test 11.00 must first be opened and saved in Service Test 11.10, before you can upgrade them to UFT.

For the tool's options, see "Batch Upgrader Command Line Options" on page 174

The following steps describe how to use the **STBatchUpgrader** tool.

- "Prerequisite" below
- "Locate the Batch Upgrader tool" below
- "Add command line options" below
- "Run the command" below

1. Prerequisite

Make sure that UFT is not running.

If you ran the upgrader tool once while UFT was running, the logs may become corrupted. Backup and delete all of the existing logs in the <Unified Functional Testing installation>\bin\logs folder before proceeding.

If desired, create a backup copy of the older tests.

2. Locate the Batch Upgrader tool

In the command line, enter the location of the STBatchUpgrader.exe file in the Unified Functional Testing/bin sub-folder.

3. Add command line options

Add the relevant options as described in "Batch Upgrader Command Line Options" on page 174, using the following syntax:

```
STBatchUpgrader.exe source [destination] [/ALM url domain project] [/login username pas
sword] [/log logfile] [/report reportfile]
```

For example, the following string runs the upgrade on all tests in the ST_11_1 folder on the ALM server, pumpkin, for the TEST1 project in the AUTOMATION domain. It places the report in c:\logs\MyLogfile.log.

```
STBatchUpgrader.exe Subject\ST11_1 /ALM http://pumpkin:8080/qcbin AUTOMATION T
EST1 /login user password  /log c:\logs\MyLogfile.log.
```

4. Run the command

Run the command. Verify the validity of the tests in the destination folder.

Reference

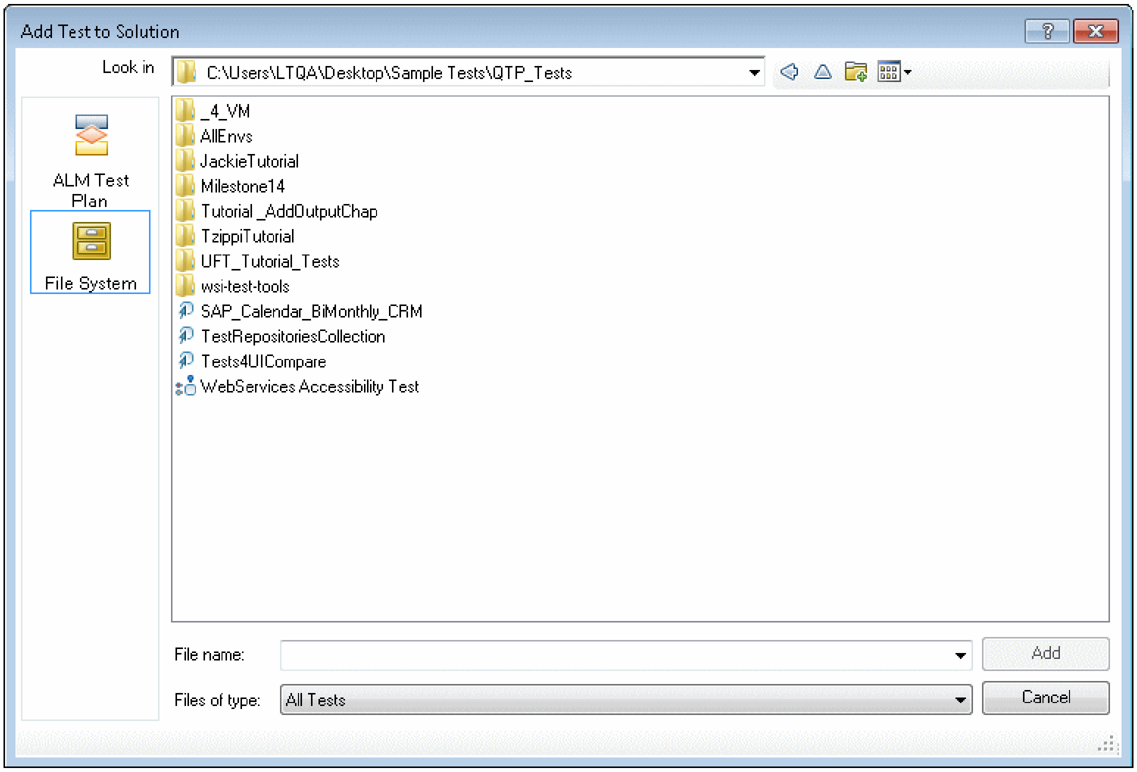
Add <Existing Document> to Solution Dialog Box

Relevant for: GUI tests and components, API testing, and business process testing

This dialog box enables you to add existing documents to a solution.

This section describes the following dialog boxes:

- Add Test to Solution dialog box
- Add Business Component to Solution dialog box
- Add an Existing Application Area to Solution dialog box



Look in: C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests

_4_VM
AllEnvs
JackieTutorial
Milestone14
Tutorial_AddOutputChap
TzippiTutorial
UFT_Tutorial_Tests
wsi-test-tools
SAP_Calendar_BiMonthly_CRM
TestRepositoriesCollection
Tests4UICompare
WebServices Accessibility Test

File name:

Files of type: All Tests

To access	Do one of the following:
	- In the "Solution Explorer Pane" (described on page 478), right-click the solution node and select **Add Existing <Document>**. - Select **File > Add Existing <Document>**. - Click the **Add** button down arrow + ▾ and select the type of document.

Important information	- Solutions are saved in your file system only and cannot be saved in an ALM project. - Using **File > Add Existing <Document>** enables you to open multiple tests, components, or application areas simultaneously instead of using **File > Open** (which closes all open documents). - If you try to open a document and there is an error with loading the test (such as ALM connectivity problems or a changed document name), an error describing the problem is displayed in the Errors pane.
Relevant tasks	"How to Create and Manage Documents" on page 136

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

UI Element	Description
<sidebar>	The location in which the document is stored, for example File System or ALM Test Plan. **Note:** You may need to select a different location before browsing to your document.
Look in	The path for the document. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the file list area to navigate to the required folder.
<file list area>	The folders and/or documents stored in the current path.
File name	The name of the documents selected in the <file list area>.
Files of type	Filters a list of displayed files to show only the file types selected. If the current folder contains other file types, they are hidden.

Add Test/Component to Solution Dialog Box

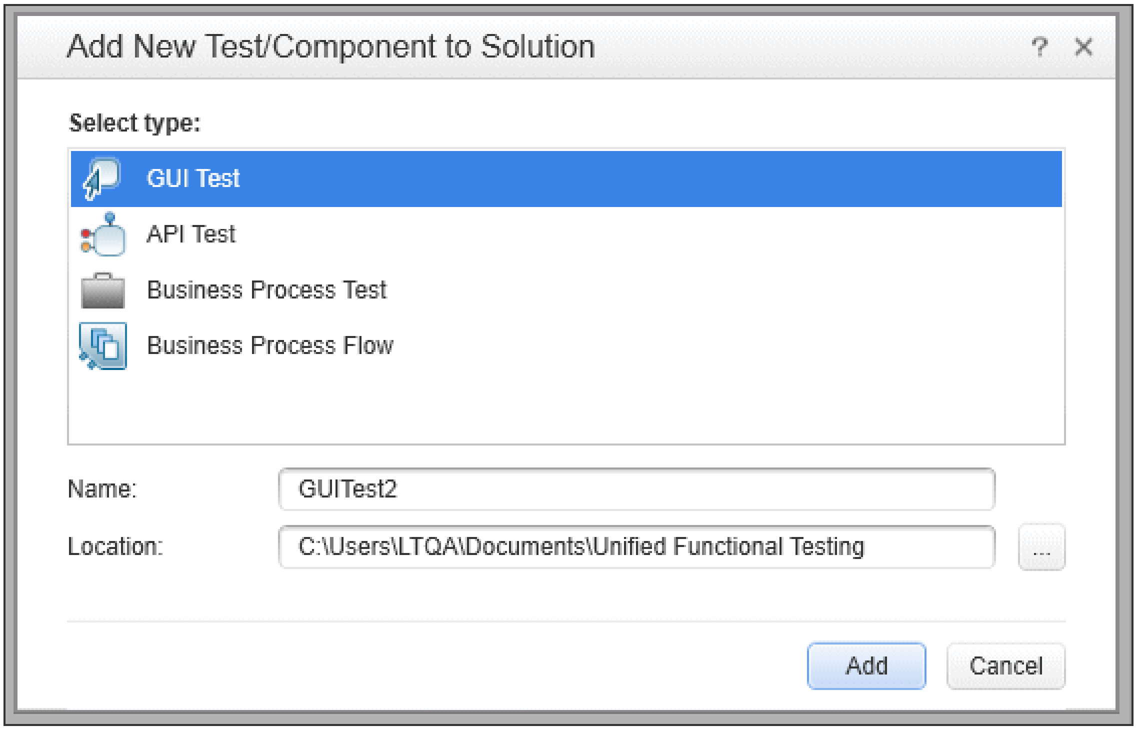
Relevant for: GUI tests and components, API testing, and business process testing

This dialog box enables you to add new tests or components to a solution.

This section describes the following dialog boxes:

- Add New Test/Component to Solution dialog box
- Add Business Component to Solution dialog box

The image below shows the dialog box to add a new test to a solution (**File > Add > New Test**).



Add New Test/Component to Solution

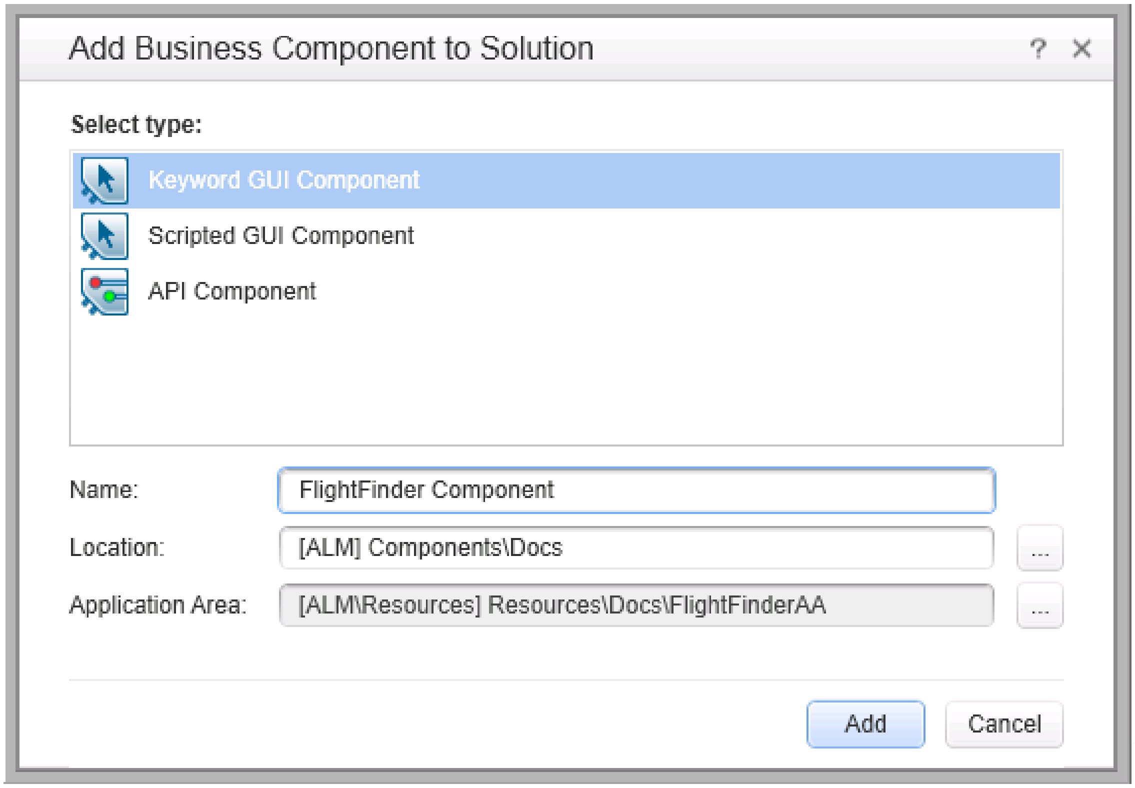
Select type:

- GUI Test
- API Test
- Business Process Test
- Business Process Flow

Name: GUITest2

Location: C:\Users\LTQA\Documents\Unified Functional Testing

The image below shows the dialog box to add a new component to a solution (**File > Add > New Business Component**).



Add Business Component to Solution

Select type:

- Keyword GUI Component
- Scripted GUI Component
- API Component

Name: FlightFinder Component

Location: [ALM] Components\Docs

Application Area: [ALM\Resources] Resources\Docs\FlightFinderAA

To access	Do one of the following: • In the "Solution Explorer Pane" (described on page478), right-click the solution node and select **Add New <Document>**. • Select **File > Add New <Document>**. • Click the **Add** button down arrow + ▾ and select the type of document.
Important information	• When you add a document to a solution, it is added to the solution in the Solution Explorer and it also opens as a new tab in the document pane. • Solutions are saved in your file system only and cannot be saved in an ALM project. For details, see "Solution Explorer Pane Overview" on page 471. • You can use this command to keep multiple tests, components, or application areas open simultaneously instead of using **File > New** (which closes all open documents). • When you save a new document in the file system, UFT suggests a default folder called **Unified Functional Testing**. For all supported operating systems prior to Windows Vista, this folder is located under your UFT installation folder. For Windows Vista and later operating systems, this folder is located under My Documents.
Relevant tasks	"How to Create and Manage Documents" on page 136

User interface elements are described below:

UI Element	Description
Select type	A list of available tests or components. For tests, you can select: • **GUI Test** • **API Test** • **API Test - Load Enabled** • **Business Process Test** • **Business Process Flow** For components, you can select: • **GUI Keyword Component** • **GUI Scripted Component** • **API Component** **Note:** The options available differ according to the license loaded and the software installed on the UFT computer.
Name	The name of the document. Use a descriptive name that helps you and others identify the document easily. Depending on the location you specify in the **Location** field, UFT suggests possible names to avoid conflicts with existing documents. **Naming conventions:** See "Troubleshooting - Naming Conventions" on page 2269.
Location	The folder of the document in the file system or an ALM project. **Note: (for ALM users)** The location of your test or component file is prefaced by an ALM path, shown as: [ALM] Components/Subject \<folder name>.
Application Area (GUI components only)	The folder location for the application area associated with your component. **Note: (for ALM users)** The file system path for your application area file is prefaced by an ALM path, shown as: [ALM\Resource] Resources\<folder name>

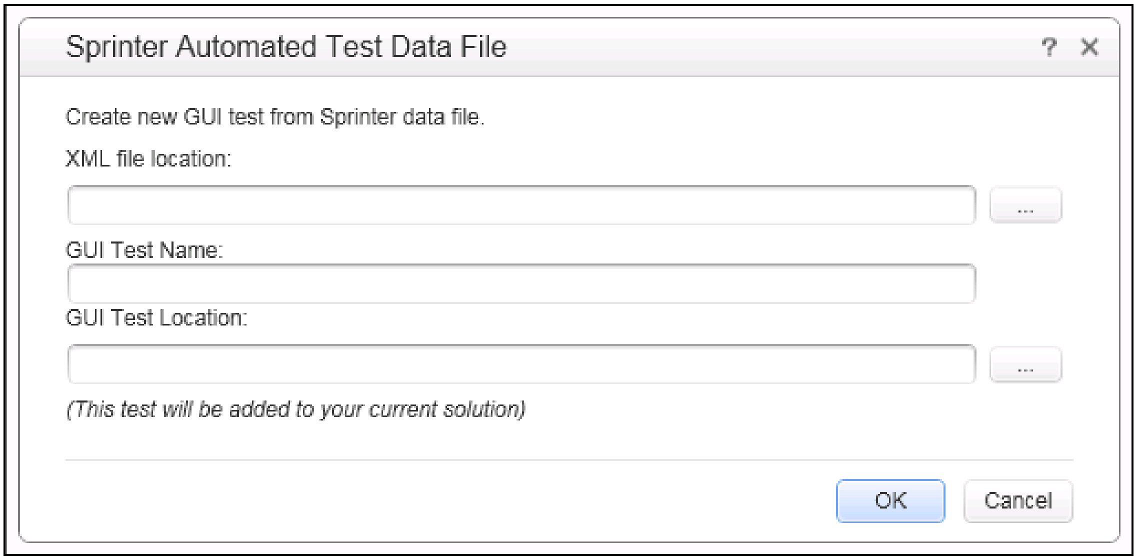
Sprinter Automated Test Data File Dialog Box

Relevant for: GUI tests only

This dialog box enables you to create or add a GUI test from an XML file containing exported Sprinter steps and test objects. UFT automatically adds this new test to the open solution.

This section describes the following dialog boxes:

- New Sprinter Automated Test Data File dialog box
- Add Sprinter Automated Test Data File (to Solution) dialog box



Sprinter Automated Test Data File

Create new GUI test from Sprinter data file.

XML file location:

GUI Test Name:

GUI Test Location:

(This test will be added to your current solution)

To access	Select **File > New > GUI Test from Sprinter Automated Test Data File** or **File > Add > GUI Test from Sprinter Automated Test Data File**.
See also	**Conceptual overview**: "Creating Tests by Importing Steps from HP Sprinter" on page 815

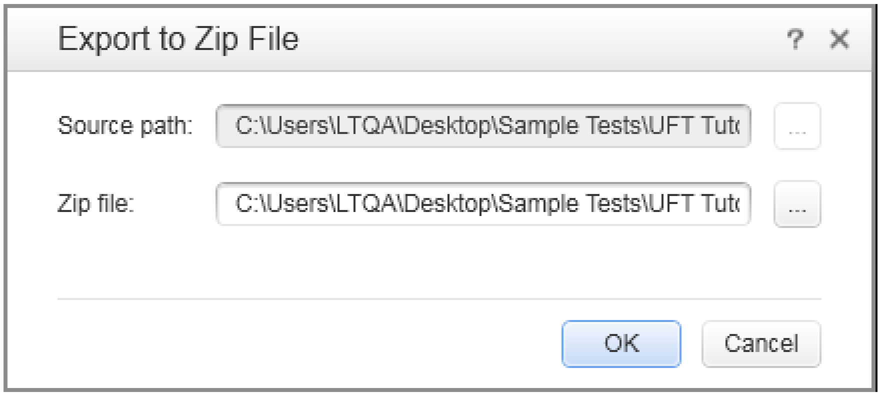
User interface elements are described below:

UI Element	Description
XML File Location	The file system path of the XML file containing the captured user actions, test objects, and comments.
GUI Test Name	The name for the new test. You can enter a new name or accept the default name.
GUI Test Location	The file system path for the new test and its files. You can enter a new path or accept the default path.

Export to Zip File Dialog Box

Relevant for: GUI tests only

This dialog box enables you to zip your GUI tests together with configuration, run-time, setup data, and (optionally) Active Screen files. Zipping these files together helps conserve space and makes tests easier to transfer.



Export to Zip File

Source path: C:\Users\LTQA\Desktop\Sample Tests\UFT Tut

Zip file: C:\Users\LTQA\Desktop\Sample Tests\UFT Tut

To access	Select **File > Export Test**.
Relevant tasks	"How to Create and Manage Documents" on page 136

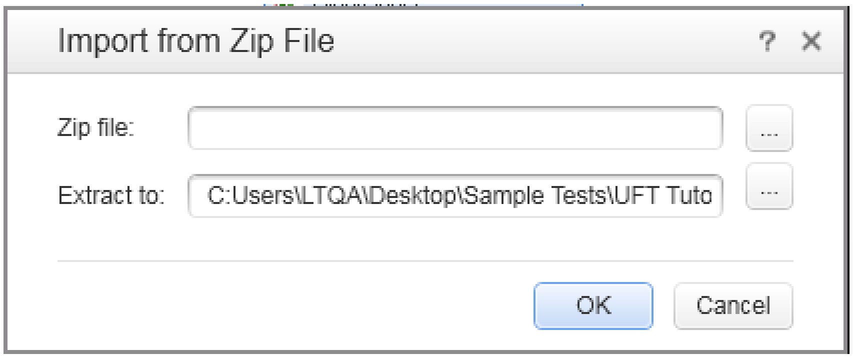
User interface elements are described below:

UI Element	Description
Source Path	The read-only current file system path of the test to be zipped.
Zip File	The file system path at which to store the zipped file. You can enter a .zip file name and path, or accept the default name and path.

Import from Zip File Dialog Box

Relevant for: GUI tests only

This dialog box enables you to extract your zipped UFT tests when needed. When the extraction process is complete, the test opens.



Zip file:

Extract to: C:\Users\LTQA\Desktop\Sample Tests\UFT Tuto

To access	Select **File > Import Test**

User interface elements are described below:

UI Element	Description
Zip File	The file path to the zipped file. You can enter a .zip file name and path, or accept the default name and path.
Extract to	The path at which to export the test and its files. You can enter the .zip file name and path, or accept the default name and path.

New <Document> Dialog Box

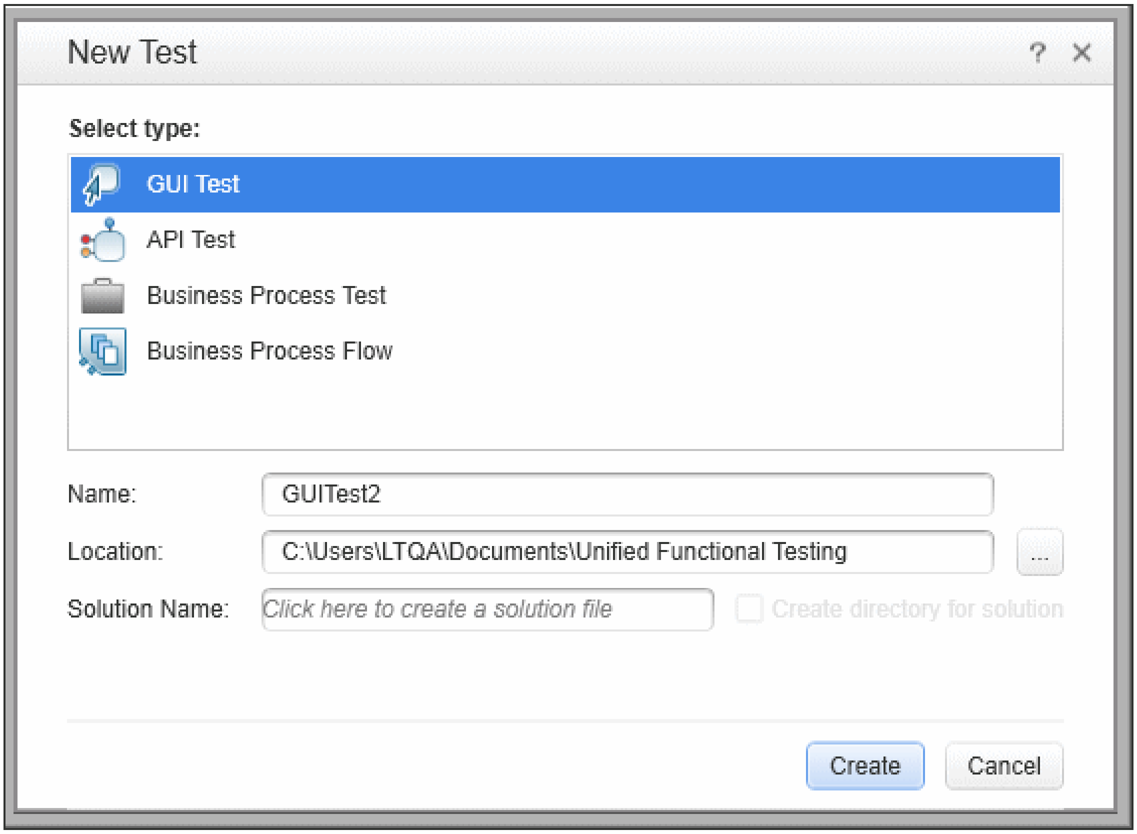
Relevant for: GUI tests and components, API testing, and business process testing

This dialog box enables you to create a new testing document.

This section describes the following dialog boxes:

- New Test dialog box
- New Business Component dialog box

The image below shows the dialog box to create a new test (**File > New > Test**).



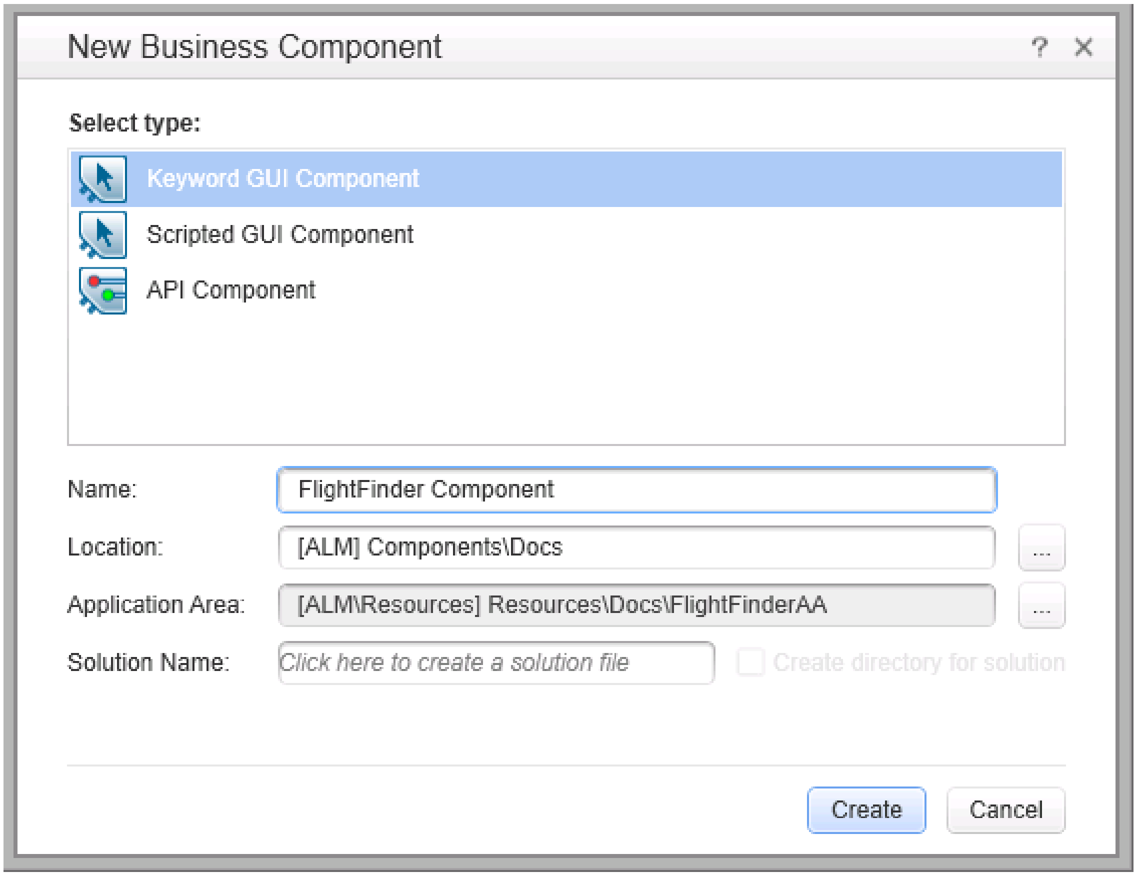
Name: GUITest2

Location: C:\Users\LTQA\Documents\Unified Functional Testing

Solution Name: Click here to create a solution file

□ Create directory for solution

The image below shows the dialog box to create a new component (**File > New > Business Component**).



Select type:

- Keyword GUI Component
- Scripted GUI Component
- API Component

Name: FlightFinder Component

Location: [ALM] Components\Docs

Application Area: [ALM\Resources] Resources\Docs\FlightFinderAA

Solution Name: *Click here to create a solution file* □ Create directory for solution

To access	1. **Prerequisite for ALM users:** Connect to your ALM project if needed. For details, see "ALM Connection Dialog Box" on page 737. 2. Do one of the following: ▪ Select **File > New > Test** (for tests only). ▪ Select **File > New > Business Component** (for components only). ▪ Click the **New** down arrow and select the type of document you want to create.

Important information	- After you create the document, it is automatically saved to the specified location. - When you save a new document in the file system, UFT suggests a default folder called **Unified Functional Testing**. For all supported operating systems prior to Windows Vista, this folder is located under your UFT installation folder. For Windows Vista and later operating systems, this folder is located under My Documents. - In ALM, components are saved in the ALM Components module. You cannot save a component directly in the root folder of this module. Create a sub-folder within it, or select an existing sub-folder.
Relevant tasks	"How to Create and Manage Documents" on page 136

User interface elements are described below:

UI Element	Description
Select type	A list of available test or component types. For tests, you can select: - **GUI Test** - **API Test** - **API Test - Load Enabled** - **Business Process Test** - **Business Process Flow** For components, you can select: - **GUI Keyword Component** - **GUI Scripted Component** - **API Component** **Note:** The options available differ depending on the license loaded and the HP applications installed on the UFT computer.

UI Element	Description
Name	The name of the document. Use a descriptive name that helps you and others identify the document easily. Depending on the location you specify in the **Location** field, UFT suggests possible names to avoid conflicts with existing documents. **Naming conventions:** See "Troubleshooting - Naming Conventions" on page 2269.
Location	The folder of the document in the file system or an ALM project. **Note: (for ALM users)** The location of your document file is prefaced by an ALM path, shown as: [ALM] Components/Subject \<folder name>.
Application Area (GUI components only)	The folder location for the application area associated with your component. **Note: (for ALM users)** The location of your application area file is prefaced by an ALM path, shown as: [ALM\Resources] Resources\<folder name>.
Solution Name	The name of the solution to create that contains your document. If you do not specify a name for a solution, a generic solution called **Solution Untitled** is created in the Solution Explorer, and the document is associated with this default solution. **Note:** All new documents must be part of a solution.

Open/New <Document>/<Resource> Dialog Box

Relevant for: GUI tests and components, API testing, and business process testing

This dialog box enables you to open an existing document or resource from the file system or your ALM project, if UFT is connected to an ALM project.

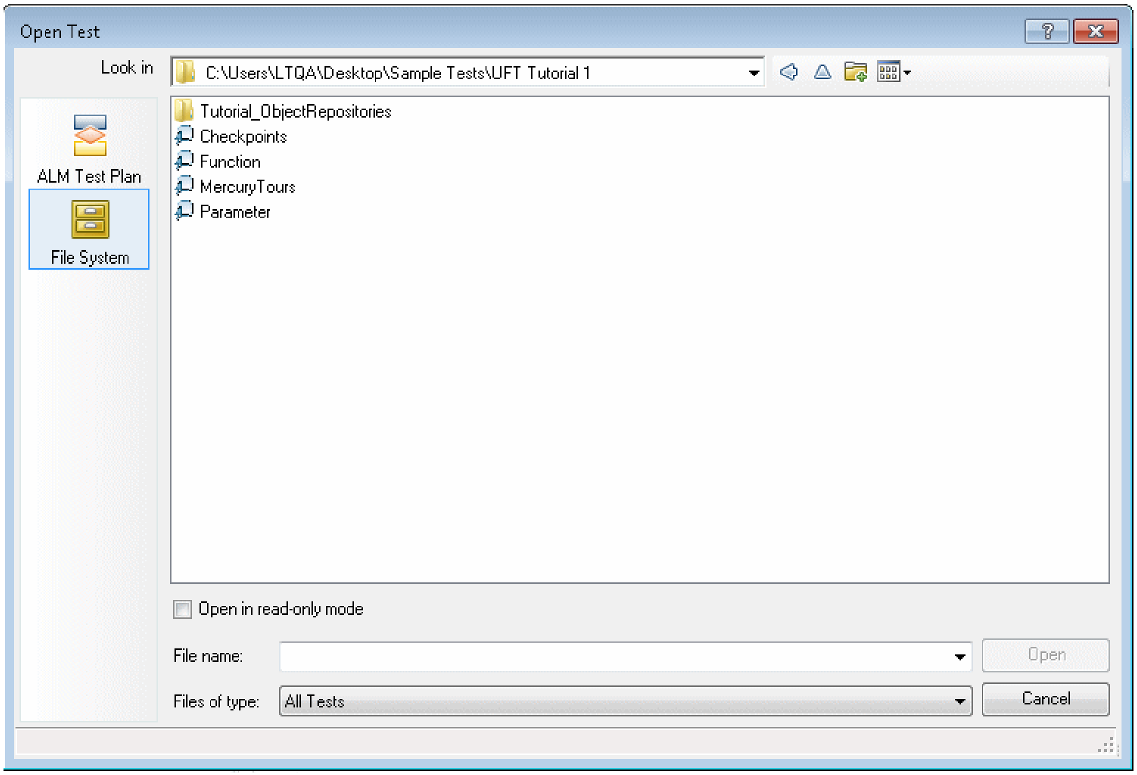
Additionally, this dialog box is used when you create a new solution, application area, or function library, enabling you to specify the location for the new item.

This section describes the following dialog boxes:

- New Function Library dialog box
- New Application Area dialog box
- New Solution dialog box
- Add New Application Area to Solution dialog box

- Open Solution dialog box
- Open Test dialog box
- Open Business Component dialog box
- Open Function Library dialog box
- Open Application Area dialog box
- Open Data Table dialog box
- Open Recovery Scenario dialog box
- Open Shared Object Repository dialog box
- Open XML File dialog box
- Open Environment Variable File dialog box

The image below shows a dialog box when opening a test. Similar options are available when opening other types of documents or resources.



Open Test

Look in: C:\Users\LTQA\Desktop\Sample Tests\UFT Tutorial 1

Tutorial_ObjectRepositories
Checkpoints
Function
MercuryTours
Parameter

□ Open in read-only mode

File name:

Files of type: All Tests

To access:

Prerequisite for ALM	Connect to your ALM project if needed. For details, see "ALM Connection Dialog Box" on page 737.
New Function Library	Click the **New** down arrow and select **Function Library.**
New Application Area	Click the **New** down arrow and select **Application Area**.
New Solution	Click the **New** down arrow and select **Solution**.
Open Test/Business Component/Application Area	Click the **Open** down arrow and select the type of document to open.
Open Function Library	Do one of the following: • Click the **Open** down arrow and select **Function Library**. • In an application area, click **Function Libraries** pane > **Add Function Library**, then click the browse button.
Open Data Table dialog box	Select **File > Settings > Resources** tab **> Other location** radio button. Then click the browse button.
Open Recovery Scenario dialog box	Do one of the following: • Select **File > Settings > Recovery** node **> Add** button, then click the browse button. • Select **Resources > Recovery Scenario Manager > Open** button. • In an application area, click **Additional Settings** pane > **Recovery** node **> Add** button, then click the browse button. For details on recovery scenarios, see "Recovery Scenarios Overview" on page 1112.
Open Shared Object Repository window	Do one of the following: • Select **Resources > Associate Repositories > Add Repository** button. • Select **Resources > Object Repository Manager**. Then, in the Object Repository Manager window, select **File > Open** or click **Open**. • In an application area, click **Object Repositories** pane > **Add Object Repository**, then click the browse button.

Open XML File dialog box	Do one of the following: - Select **Design > Checkpoint > XML Checkpoint (From Resource)**. Then, in the "XML Source Selection - Checkpoint / Output Value Properties Dialog Box" (described on page 1482), select **Create checkpoint from XML file** and click **Browse**. - Select **Design > Output Value > XML Output (From Resource)**. Then, in the "XML Source Selection - Checkpoint / Output Value Properties Dialog Box" (described on page 1482), select **Create output value step from XML file** and click the browse button.
Open Environment Variable File dialog box	1. Select **File > Settings > Environment** node **> User-defined** variable type. 2. Select the **Load variables and values from external file** check box and click the browse button.

Important information	• When you create a new document in the file system, UFT suggests a default folder. For all supported operating systems prior to Windows Vista, this folder is located under your UFT installation folder. For Windows Vista and later operating systems, this folder is located under **My Documents**. • After you create the document, it is automatically saved in the specified location. • When you open a document in read-only mode, the title bar displays **Read Only** and all tabs associated with that document (for example, actions in a test) display a lock icon. • Each time you open a document using **File > Open**, it replaces and closes any documents or solutions currently in use. • You can also open a recently used test or component by selecting from the **Recent** files in the **File** menu. If you select a test or component when you are not connected to an ALM project, or if you select a component that is stored in a different ALM project, UFT displays a message asking you if you want to connect to that project. • If you try to open a document and there is an error with loading the test (such as ALM connectivity problems or a changed document name), an error describing the problem is displayed in the Errors pane. • If your GUI test has locked resources, see "Open Test with Locked Resources - Message Box" on page 163. • **For BPT in UFT:** The first time that you open a business process test or flow from UFT, you must connect as a user with administrator privileges on the computer on which you are connecting. • When trying to open a test saved in Service Test, version 11.10 or 11.20, UFT prompts you to upgrade the test using the Batch Upgrader tool (described on page 143). If the test or component was created in Service Test 11.00, you must first open and save it in Service Test 11.10 before you can upgrade it using the Batch Upgrader. • If you try to open a test or component last saved in a version of QuickTest earlier than 9.5, an error message is displayed in the UFT Error Pane. You must first open it in QuickTest 10.00 or 11.00 to upgrade it. Assets last saved in QuickTest 10.00 or later can be opened directly in UFT.
Relevant tasks	"How to Create and Manage Documents" on page 136

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

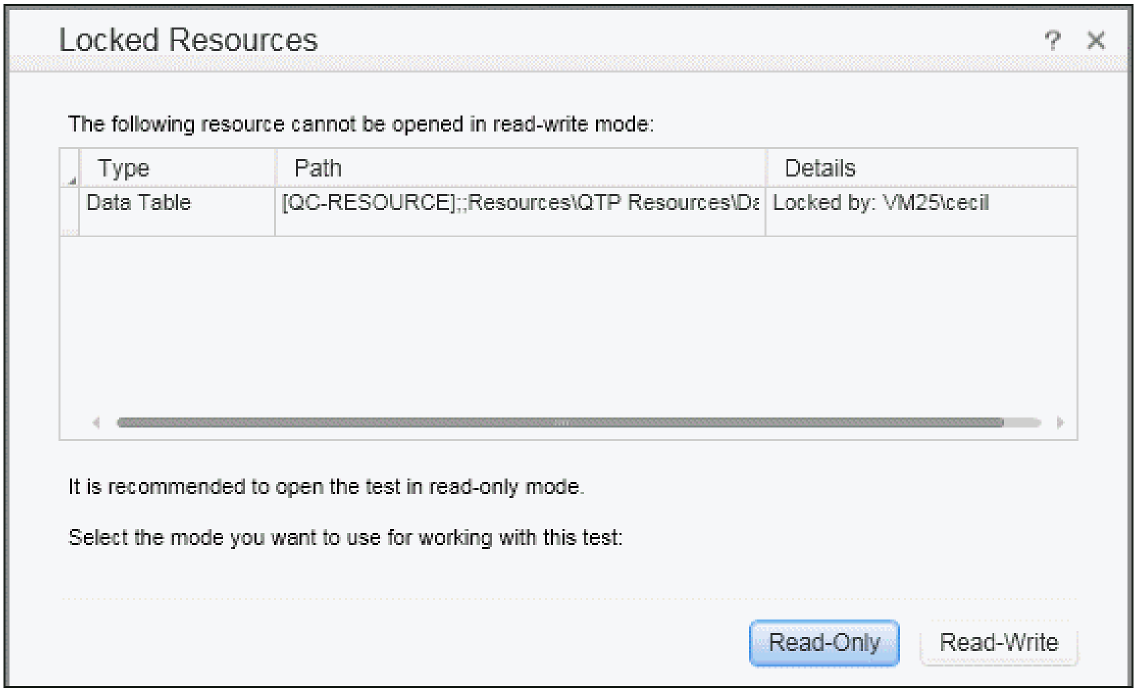
UI Element	Description
<sidebar>	The location in which the document is stored, for example File System, ALM Test Plan, or ALM Resources. **Note:** • You may need to select a different location before browsing to your document. • Tests stored in ALM are always stored in ALM Test Plan. Business components are always stored in ALM Components. Application areas are always stored in ALM Resources.
Look in	The path for the document. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the file area to navigate to the required folder.
<file list area>	The folders and/or documents stored in the current path. **Note: (for ALM users)** If the document is stored in an ALM project with version control support, you can view version control information for the document by clicking the **Views** down arrow and selecting **Details**. For details, see "Checking Assets Out of the Version Control Database" on page 797.
File name	The name of the document selected in the file list area.
Files of type	Filters the list of displayed folders or documents to display the selected file type. If the current folder contains other file types, those files are hidden.

UI Element	Description
Open in read-only mode	Opens the document in read-only mode. This option enables you to view the document but not modify it. **Note: (for ALM users)** A document also opens in read-only mode if: • You opened a document that is currently checked in to the version control database (for projects that support version control). • You opened a document that is currently checked out to another user (for projects that support version control). • You opened a document from an earlier version of ALM and the document has not yet been updated to the current format. For details, see "Checking Assets Out of the Version Control Database" on page 797.
Open	Opens the selected document.

Open Test with Locked Resources - Message Box

Relevant for: GUI tests only and API tests

This message box opens when you try to open a document that is associated with a locked resource file, such as a locked data table or object repository file.



The following resource cannot be opened in read-write mode:

Type	Path	Details
Data Table	[QC-RESOURCE];;Resources\QTP Resources\Da	Locked by: VM25\cecil

It is recommended to open the test in read-only mode.

Select the mode you want to use for working with this test:

To access	Open a document with a locked resource file, such as a locked data table or shared object repository.
Important information	When you try to open a document with locked resource files, the locked resource files open in read-only mode. You can open the document in "Read-Only" or "Read-Write" mode.
See also	"Opening and Saving Tests with Locked Resources" on page 134

User interface elements are described below:

UI Element	Description
Read-Only	Opens the document in read-only mode, enabling you to run it but not make any changes to it. All editing options are disabled and you cannot edit the document. **Note:** Even though the document itself cannot be saved, if you run the document and choose to save the results, the results file is saved. You can select the **Save As** option from the **File** menu, save the document under a new name, and then edit the document. The same is true for a shared object repository file opened in read-only mode.
Read-Write	Opens the document in read-write mode, enabling you to run and edit it, and save any changes you make to it. However, any changes to referenced and locked UFT files cannot be saved. Locked resource files are opened in read-only mode even if the referencing document is opened in read-write mode. **Example:** Suppose you open a test in read-write mode that has a locked data table associated with it. When you edit the test, any changes to the data table are not saved. If any of the data table values change as a result of editing the test, and you save the changes to the test, the test may fail the next time it calls the unmodified data table values. It is therefore recommended to open the test in read-only mode or to close the test without saving.

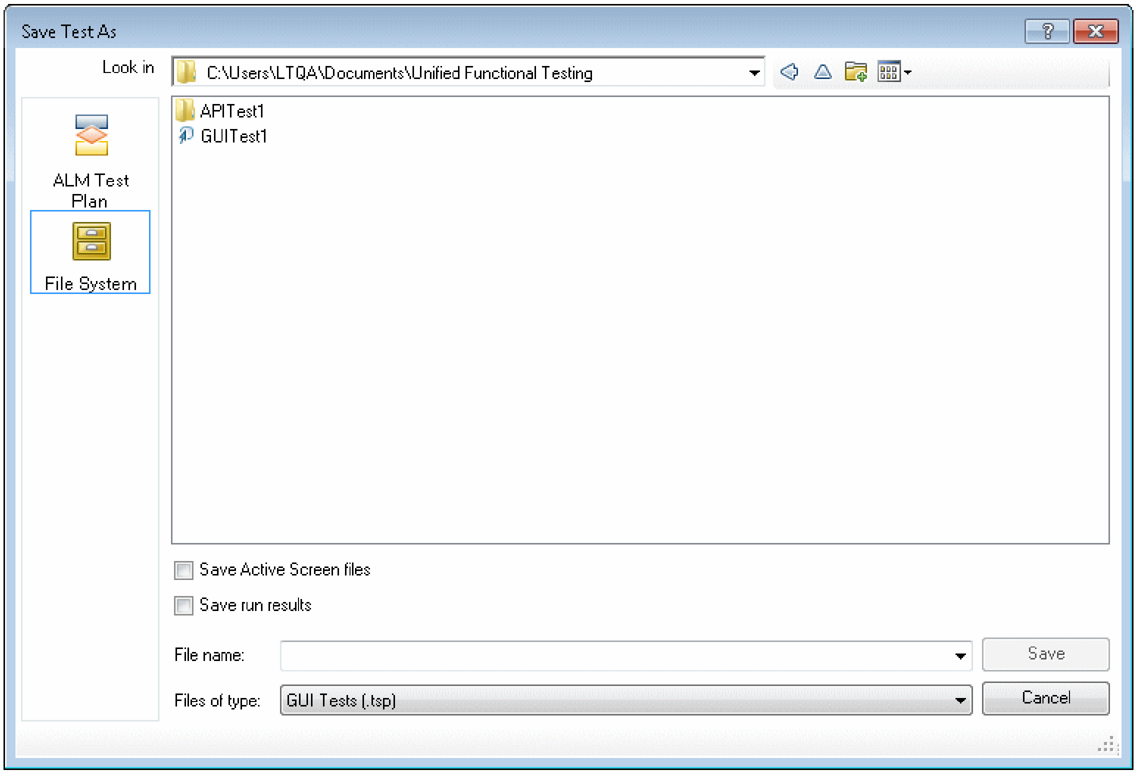
Save <Resource>/Save <Document> As Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to save a new resource, or save a copy of an existing document with another name. You can save the document or resource in the file system or your ALM project, if UFT is currently connected to an ALM project.

This section describes the following dialog boxes:

- Save Function Library dialog box
- Save Shared Object Repository dialog box
- Save Recovery Scenario dialog box
- Save Environment Variable File dialog box
- Save (Test/Business Component/Application Area/Function Library) As dialog box



To access	
Prerequisite for ALM	Connect to your ALM project if needed. For details, see "ALM Connection Dialog Box" on page 737.
Save Function Library	In an application area sidebar, click **Function Libraries** and then click **Create Function Library** ✱.
Save Shared Object Repository	Do one of the following: • In the Object Repository Manager, select **File > Save/Save As** when working with an object repository. • In an application area sidebar, click **Object Repositories** and then click **Create Object Repository** ✱.
Save Recovery Scenario	In the Recovery Scenario Manager, click the **Save** button after creating a recovery scenario.
Save Environment Variable File	In the Test Settings dialog box, select an environmental variable and click **Export**.

Save As (for tests, and business components, application areas, and function libraries)	Do one of the following: • In the document pane, right-click the document tab, and select **Save <document name> As**. • Select **File > Save <document name> As**. • For API testing only: **File > Save (Other) > Save <component name > As Business Component**. • For GUI testing only: Select **Save** or click the **Save** button 💾 in the resource dialog box
Important information	• You must use the **Save As** option in UFT to save a document with another name or create a copy of a document. You cannot copy a document or change its name directly in the file system or in ALM. • If changes are made to an existing document, an asterisk (*) is displayed in the title bar until the document is saved. • For GUI components: If you want to associate a resource file with an application area, and take advantage of the Resources and Dependencies model, save the resource file to the ALM Test Resources module in an ALM project. For details, see "Resources and Dependencies Model Terminology" on page 759.
Relevant tasks	"How to Create and Manage Documents" on page 136

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

UI Element	Description
<sidebar>	The location in which to save the document or resource, for example, File System or ALM Test Plan. **Note:** For business components the location is always ALM Components, and for and application areas it is always ALM Resources. For business process tests or flows, the location is always ALM Test Plan.
Look in	The path for the folder in which to save the document or resource. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the file area to navigate to the required folder. **Note: (for application areas only)** You must save application areas in the default location. If you need to move the application area after saving it, you can do that from within ALM, as described in the *HP Application Lifecycle Management User Guide*.

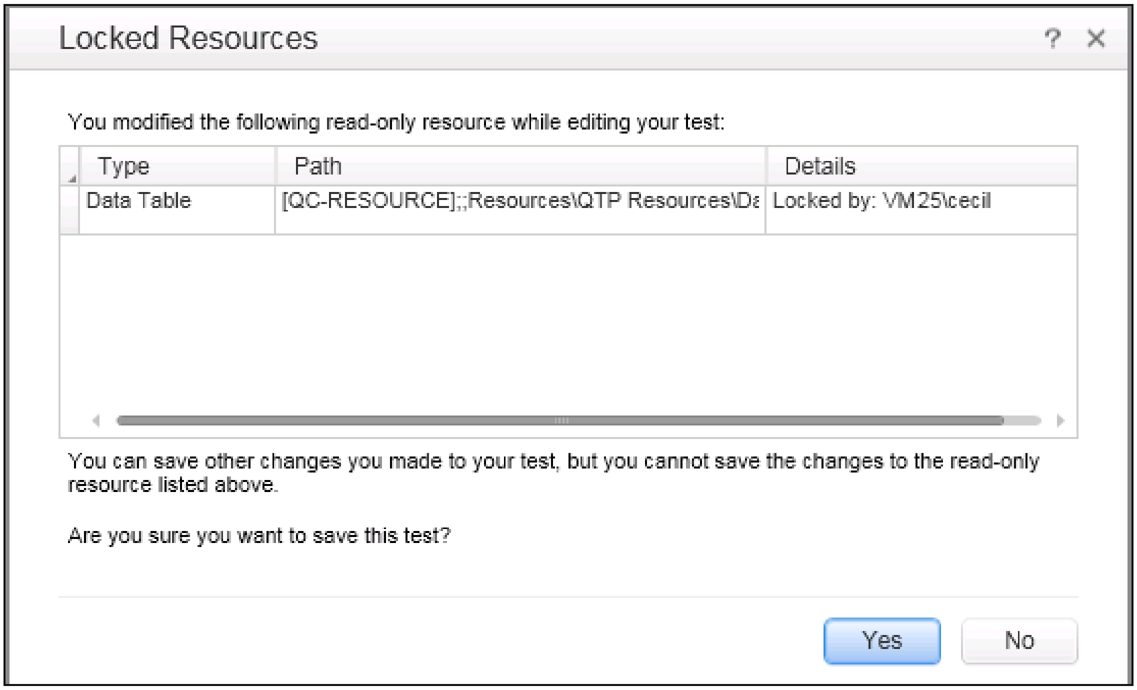
UI Element	Description
<file list area>	The folders and/or documents or resource files stored in the current path. **Note:** (**for ALM users)** If the document is stored in an ALM project with version control support, you can view version control information for the document by clicking the **Views** down arrow and selecting **Details**. For details, see "Checking Assets Out of the Version Control Database" on page 797.
File name	The name of the document or resource. Use a descriptive name that helps you and others identify the document or resource easily. **Naming conventions:** See "Troubleshooting - Naming Conventions" on page 2269
Files of type	Filters the file types displayed in the file list area. You can choose to display a specific type of file or all file types.
Save Active Screen files (GUI tests and scripted GUI components only)	Saves the Active Screen files with your test or scripted component. If you clear this check box, your Active Screen files will not be saved, and you will not be able to edit your test or scripted component using the options that are normally available from the Active Screen. **Tip:** • If you are using the test or scripted component *only* for run sessions, and if you want to conserve disk space after you finish designing the test or scripted component, clear the **Active Screen files** check box. • If you had previously cleared the Save Active Screen files check box, and now want to edit your test or scripted component using Active Screen options, regenerate the Active Screen information by performing an **Update Run** operation. For details, see "How to Update Test Object Descriptions, Checkpoints, or Output Values, or Active Screen Captures" on page 1088. • To set Active Screen preferences, such as instructing UFT not to capture Active Screen files while recording or to only capture Active Screen information under certain conditions, you can modify the options in the **Active Screen** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Active Screen** node). For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547. For details on the Active Screen and its uses, see "Active Screen Overview" on page 194.

UI Element	Description
Save run results (GUI tests only)	Saves any existing run results with your document. **Note:** The check box is available only for documents that are saved in the file system, and that ran at least once. If you clear this check box, your run result files are not be saved, and you will not be able to view them later. **Tip:** To conserve disk space if you do not require the run results for later analysis, or if you are saving an existing document under a new name and do not need the run results, clear the **Save run results** check box. For details on run results, see the *HP Run Results Viewer User Guide*.

Save Test with Locked Resources - Message Box

Relevant for: GUI tests and API tests

This message box opens when you save a read-write document with locked resource files, which have changed since you opened the associated document.



You modified the following read-only resource while editing your test:

Type	Path	Details
Data Table	[QC-RESOURCE];;Resources\QTP Resources\Da	Locked by: VM25\cecil

You can save other changes you made to your test, but you cannot save the changes to the read-only resource listed above.

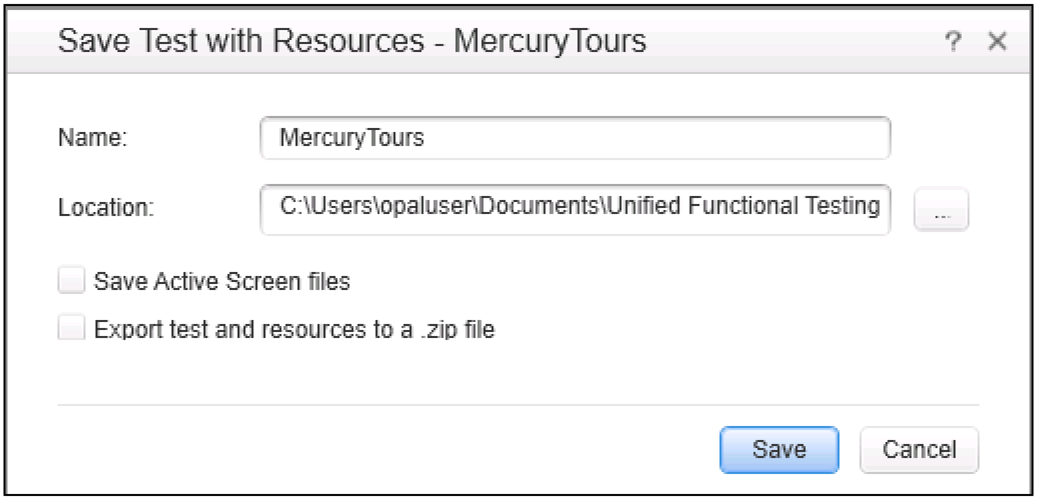
Are you sure you want to save this test?

To access	Save a document with a locked resource file that has been modified during the current UFT session.
Important information	It is recommend to avoid saving documents when locked resource fields have been modified. If you do save the document, the next time you run the test or component, the run may fail because of unexpected resource file values.
See also	"Opening and Saving Tests with Locked Resources" on page 134

Save Test with Resources Dialog Box

Relevant for: GUI tests

This dialog box enables you to save a fully portable copy of a test and its resource files to a local drive or other storage device, eliminating the need for network or ALM connections.


Save Test with Resources - MercuryTours

Name: MercuryTours

Location: C:\Users\opaluser\Documents\Unified Functional Testing

□ Save Active Screen files

□ Export test and resources to a .zip file

To access	Select **File > Save (Other) > Save <test name> with Resources**.

Important information	**Before you create a copy of the test:** • Resolve any missing resources. • Save the original test. • Make sure that the test and its resources are all updated to the current UFT version. • Make sure that all files associated with the source test are editable. • Make sure you have write permissions for the folder in which you want to create a copy of the test. **After you make a copy of the test:** • A report is generated in HTML format, listing: ▪ The name of the test, the name of the user that saved this copy of the test, and the date on which the test was copied. ▪ A record for each resource that was copied with the test, specifying: ○ the name of the resource ○ the type of resource (for example, function library, an external data table, or shared object repository) ○ the path from which the resource was copied-- the status of the copied resource (for example, the resource was saved successfully) ○ the current location of the copied resource You can open the HTML report from the copied test's root folder. • The copied test becomes the active test in the UFT window. • All links to the source files are severed. Therefore, any modifications you make to the copied test are applied only to the copied test. • Any external actions associated with your test are saved as internal actions when saving a test with resources.
Relevant tasks	"How to Create and Manage Documents" on page 136
See also	"Portable Copies of Tests" on page 133

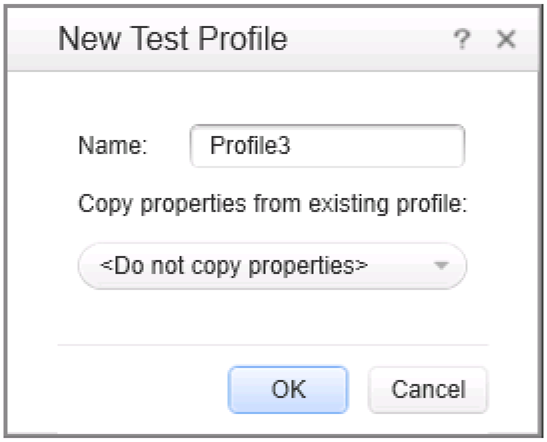
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
Name	The name for the saved test. **Naming conventions:** See "Troubleshooting - Naming Conventions" on page 2269
Save Location	Specifies the root folder in which to save the test. By default, for any operating system prior to Windows Vista, the root folder is <UFT installation folder>\Tests. For Windows Vista and later operating systems, the root folder is:...\MyDocuments\HP\Unified Functional Testing. However, you can specify any folder on a local, network, or portable drive. **Note:** The folder you specify must not already contain a sub-folder with the same name as the test.
Save Active Screen files	Instructs UFT to save any existing Active Screen files with your test. Clearing the Save Active Screen files check box can be especially useful for conserving disk space if you have finished designing the test and are using the test only for run sessions. **Note:** If you clear this box, your Active Screen files are not copied over with the test and its resources, and you are not able to edit your test using the options that are normally available from the Active Screen. **Tip:** If you clear the Save Active Screen files check box and later want to edit your test using Active Screen options, you can regenerate the Active Screen information by performing an Update Run operation. For details, see "How to Update Test Object Descriptions, Checkpoints, or Output Values, or Active Screen Captures" on page 1088.
Archive test and resource files in a .zip file	Creates a .zip file of the test and its resources, and stores the .zip file in the folder you specified in the Save Location box.

New Test Profile Dialog Box

Relevant for: API testing only

This dialog box enables you to create a new profile for your test with pre-defined user variables.



New Test Profile

Name: Profile3

Copy properties from existing profile:

<Do not copy properties>

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Select the **Start** or **End** step in the canvas. 3. In the Properties pane, open the **User Variables** view . 4. Click the **Add Profile** button .
Relevant tasks	"How to Define API Test Properties or User/System Variables" on page 141

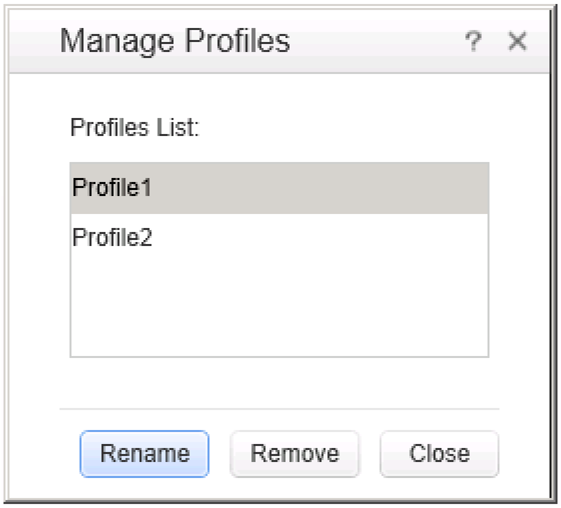
User interface elements are described below:

UI Element	Description
Name	A name for the profile as it will appear in the profile list drop down list.
Copy properties from existing profile	A drop down list of existing profiles. **Default:** Do not copy properties, creates a new empty profile with the user variables of the active profile, without values.

Manage Profiles Dialog Box

Relevant for: API testing only

This dialog box enables you to remove and rename user variable profiles.



To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Select the **Start** or **End** step in the canvas. 3. In the Properties pane, open the **Test Variables** view 🌐. 4. Add additional profiles using **Add Profile**. 5. Click the **Manage Profiles** button.
Relevant tasks	"How to Define API Test Properties or User/System Variables" on page 141

User interface elements are described below:

UI Element	Description
Profiles List	A list of all the profiles defined for this test.
Rename	Allows you to rename the selected profile.
Remove	Deletes the selected profile.

Batch Upgrader Command Line Options

Relevant for: API testing only

You operate the batch upgrade tool from the command line. For task details, see "How to Upgrade API Tests Using the Batch Upgrader " on page 143. The following table describes the command line options:

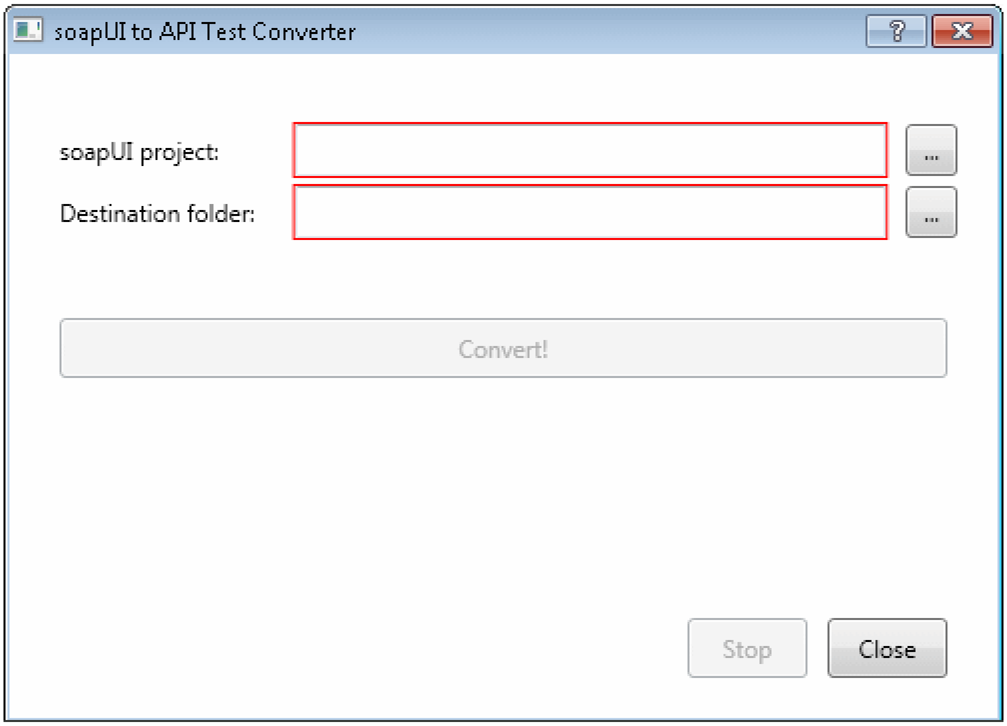
UI Elements	Description
/ALM	Indicates that ALM connection information will follow.
/log	Indicates that the log will be written to an alternate file. By default, log files are store in the <UFT installation>\bin\logs folder.
/login	Indicates that login information will follow.
/report	Instructs the upgrader tool to generate a summary report.
destination	• For tests on the File system: The full UNC path of the folder in which to store the tests after the upgrade. • For tests in ALM: a target folder path under the Test Plan module, in which to store the upgraded tests. **Note:** • The **destination** value must be a local or remote folder with the same write access permissions as the **source** path. • The destination folder should be an empty folder that does not include any tests. • If you do not specify a destination folder, the tests will be upgraded in the source folder, overwriting the originals.
domain	The name of an ALM domain in which the source tests are stored.
logfile	The full path of the file to which the log will be written.
project	The name of an ALM project in which the source tests are stored.
reportfile	The full path of an existing folder, with the file name including an extension, to where the summary report should be written. For example, c:\logs\MySummary.txt.

UI Elements	Description
source	• **For tests on the File system:** The full UNC path of the folder containing the tests to be upgraded. • **For tests in ALM:** a source folder path under the Test Plan module).
url	The URL of an ALM instance to which to connect in the following form: http://{instance_domain}:8080/qcbin
username, password	For ALM mode, the username and password with which to log in to the ALM project.

soapUI to API Test Converter

Relevant for: API testing only

Using the soapUI Converter tool, you can convert soapUI tests to UFTAPI tests.



To access	Do one of the following:
	• Select **soapUI to API Test Converter** from the product's **Tools** section under the **Start** menu. • Run the soapUI2APITest.exe file from the product's bin folder.

User interface elements are described below:

UI Element	Description
soapUI project	The absolute path of the .xml file representing the soapUI project.
Destination folder	The target folder in which to store the converted test.
Convert	Begins the conversion process.

Command Line Syntax

You can also run the soapUI converter utility from the command line. Run the following command in the command line:

```
soapUI2APITestCMD.exe /<source soapUI_file> /destination <destination directory>/
logs <log_directory>
```

Command line options are described below:

Command Line Switch	Description
/source	The absolute path to the soapUI file with an .xml extension, to be converted.
/destination	The absolute path of the folder to where the created API tests will be written.
/logs (optional)	The absolute path of the folder in which to write the log file. If this option is omitted, the log file is written to the destination folder.
-? or /?	Show the parameters and their usage. For example: soapUI2APITestCMD.exe -?

Troubleshooting and Limitations - UFT Testing Documents

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitation for working with testing documents:

Troubleshooting and Limitations - Opening and Saving Tests

Relevant for: GUI tests and components and API testing

This section describes general troubleshooting and limitation for creating, opening and saving testing documents, as well as opening tests or components last modified in previous versions of QuickTest or Service Test.

Folder names

It is not possible to create an API test in a path containing a **'='** character.

Opening documents stored in ALM

Opening a document from the Recent Files.

You can open components from the **Recent** files list in the **File** menu. If you select a test located in an ALM project, but UFT is not currently connected to the correct project, the Connect to ALM Project dialog box opens and displays the correct server, project, and the name of the user who most recently opened the test or component on this computer.

This dialog box also opens if you choose to open a test or component that was last edited on your computer using a different ALM user name. You can either log in using the displayed name or you can click **Cancel** to stay logged in with your current user name.

Opening a document that was last modified in an earlier version of QuickTest.

If a test is stored in ALM and was last modified using a version of QuickTest earlier than 9.5, it opens in read-only mode. To edit the test, it must be upgraded to QuickTest 11.00 using the QuickTest Asset Upgrade Tool for ALM (found on the QuickTest 11.00 installation DVD). After upgrading the test to QuickTest 11.00, you can open it in UFT.

Opening a solution containing test documents stored in ALM.

Testing documents that are part of the same solution cannot be stored in different ALM projects, domains, or servers.

Opening documents stored in the file system

UFT does not support hidden files. If you mark a UFT test, or other UFT file as hidden (by selecting the **Hidden** attribute), UFT may behave unexpectedly.

Opening a document that was last modified in an earlier version of QuickTest.

- If you try to open a test or component last saved in a version of QuickTest earlier than 9.5, an error message is displayed in the UFT Error Pane. You must first open it in QuickTest 10.00 or 11.00 to upgrade it. Assets last saved in QuickTest 10.00 or later can be opened directly in UFT.

- If the test contains objects in the local object repositories of one or more actions in the test, the relevant add-in must be installed to convert the test to the current format. Otherwise, the test opens in read-only format.

- If you choose to convert the test, it is updated to the current format and you can modify it as needed. If you save the converted test, it cannot be used with earlier versions of QuickTest.

- If you choose to view the test in read-only format, it appears as it did previously, using all of its original settings, but you cannot modify it.

- If you have many tests that need to be updated to the current format, you can create a script that iterates through all of your tests to open and save each one in the new format. For details on creating scripts, see "UFT Automation Object Model Overview" on page 1156.

  To view a sample script that converts old tests to the current version, see the **Convert a Set of Tests from an Older QuickTest Version to the Current Version** example in the *HP UFT Automation Object Model Reference* (**Help > HP UFT GUI Testing Advanced References Help > HP UFT Automation Object Model Reference**).

Opening a document that was last modified in an earlier version of Service Test.

When trying to open a test saved in Service Test, version 11.10 or 11.20, UFT prompts you to upgrade the test using the Batch Upgrader tool (described on page 143). If the test or component was created in Service Test 11.00, you must first open and save it in Service Test 11.10 before you can upgrade it using the Batch Upgrader.

Opening and saving tests with resources

- If a test has locked resource files, you can open the test in read-only mode or in read-write mode. In all cases, the locked resource file opens in read-only mode.

  For details, see "Opening and Saving Tests with Locked Resources" on page 134.

  For a description of the displayed message boxes, see "Open Test with Locked Resources - Message Box" on page 163 and "Save Test with Locked Resources - Message Box" on page 168.

Opening and saving documents in Windows 7

- The Libraries feature of Windows 7/Windows 8 is not supported. If you attempt to open a test from a library location or save a test to a library location, UFT may behave unexpectedly.

Workaround: Browse to your saved tests using standard file paths, even if they are included in a library, such as the Documents library.

For example, to browse for a test that is stored in the default UFT folder, use:

```
Computer > C: > Program Files > HP Software > HP Unified Functional Testing> Tests > <Tes
tName>
```

Instead of:

```
Libraries > Documents > HP Unified Functional Testing > <TestName>
```

- If you work with UFT on a Windows Vista, Windows 7, or Windows 8 operating system with User Account Control (UAC) enabled, and you open a test from a protected location (such as Program Files), it is opened in read-only mode and a message is displayed that you do not have permissions to open it in read-write mode.

Unexpected read-only files

Documents that were saved in the file system in system folders like (%Windir% or Program files) while UAC was disabled can be opened only in read-only mode when the UAC is set to ON.

Workaround: Copy the asset to another location and open the files from the new location.

Multilingual support

Names and paths of tests and resources (for example, function libraries, object repositories, and recovery scenarios) are not Unicode compliant and therefore should be specified in English or in the language of the operating system.

Troubleshooting and Limitations - Upgrading Service Test Tests

Relevant for: API testing only

This section describes limitations for opening tests last modified with earlier versions of Service Test.

- When upgrading a test from version 11.00 (via 11.10), Service Test does not retain the Security settings.

  Workaround: In Service Test 11.00, save the Security Scenario to an .stss file. Import this file for your service when you upgrade it. For details, see "Setting Security Overview" on page 1878.

- Tests last modified in Service Test 11.20 that contain a Call QuickTest Professional Test step, cannot be opened in UFT.

- In certain instances, if the upgrader was unable to upgrade the test, you may need to modify the code to make it compatible with the current version:

  - The user code file is now titled TestUserCode.cs.

  - The namespace at the beginning of the test is Script.

  - The class definition is public class TestUserCode : TestEntities.

    ```
    namespace Script
    {
           using System;
           using System.Xml;
           using System.Xml.Schema;
           using HP.ST.Ext.BasicActivities;
           using HP.ST.Fwk.RunTimeFWK;
           using HP.ST.Fwk.RunTimeFWK.ActivityFWK;
           using HP.ST.Fwk.RunTimeFWK.Utilities;
           using HP.ST.Fwk.RunTimeFWK.CompositeActivities;
           using System.Windows.Forms;
           using HP.ST.Ext.FTPActivities;


           [Serializable()]
           public class TestUserCode : TestEntities
           ...
    ```

  - The args variable is no longer supported.

    Workaround: Change the args variable name to the concrete Activity name as it appears at the top of the Properties pane. For example, change args.Output.outStr = args.Input.inStr; to CodeActivity8.Output.outStr=CodeActivity8.Input.inStr.

  - The soapUI to API Test Converter tool does not support links between steps. If you convert a soapUI test with linked Web service calls, the links will not be transferred to the API/Service Test test.

Part 2: UFT Panes

Chapter 4: UFT Window Layout

Relevant for: GUI tests and components and API testing

This chapter includes:

Concepts

UFT Window Layout Customization - Overview

Relevant for: GUI tests and components and API testing

You can modify the layout of the UFT window. For example, you can move and resize panes, select to show or auto-hide panes, create tabbed panes, and select which toolbars to display. If needed, you can also restore the default layout.

You can also resize the UFT window to suit your needs for each type of UFT session (view/edit, record, and run sessions). For example, you can display UFT in full view when creating or editing a test, component, or application area, and minimize the UFT window during a run session.

When you customize or restore the UFT window layout, UFT applies the changes to all document types and session types.

UFT Window Layout Customization in Different Modes

UFT works in several different modes: view/edit, run, and record (for GUI testing only). You may want to modify the UFT layout to match the functionality of a mode. For example, when recording a GUI test or component, it is often convenient to have UFT partially visible. This enables you to watch steps being added as you record your test without viewing the Active Screen. In addition, when running a test or component, it is often convenient to minimize UFT so that you can view your application during the run session. When viewing or editing a test or component, it may be convenient to maximize the UFT window, with all panes showing.

UFT remembers the size and location of its main window and all of its panes for each mode. When UFT enters a mode, the layout reverts to the most recently used layout for that mode. This means that the main UFT window and each of its panes are maximized, minimized, or resized, based on the previous layout of the current mode.

Tasks

How to Customize the UFT Window

Relevant for: GUI tests and components and API testing

This section describes how to customize and modify the UFT window layout and panes.

This task includes the following steps:

- "Move Panes" below
- "Show and Hide Panes" on page 188
- "Float and Dock Panes" on page 189
- "Set the UFT Layout for Record Mode" on page 190
- "Set the UFT Layout for Run Mode" on page 190
- "Restore the Default Layout of the UFT Window" on page 190

Move Panes

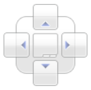
You can move the UFT window panes to suit your own personal preferences. You can rearrange the panes, and you can also change a pane to a tabbed pane, and vice versa.

1. In the UFT window, drag the title bar or tab of the pane you want to move. (If the required pane is not displayed in the UFT window, you can select it from the **View** menu).

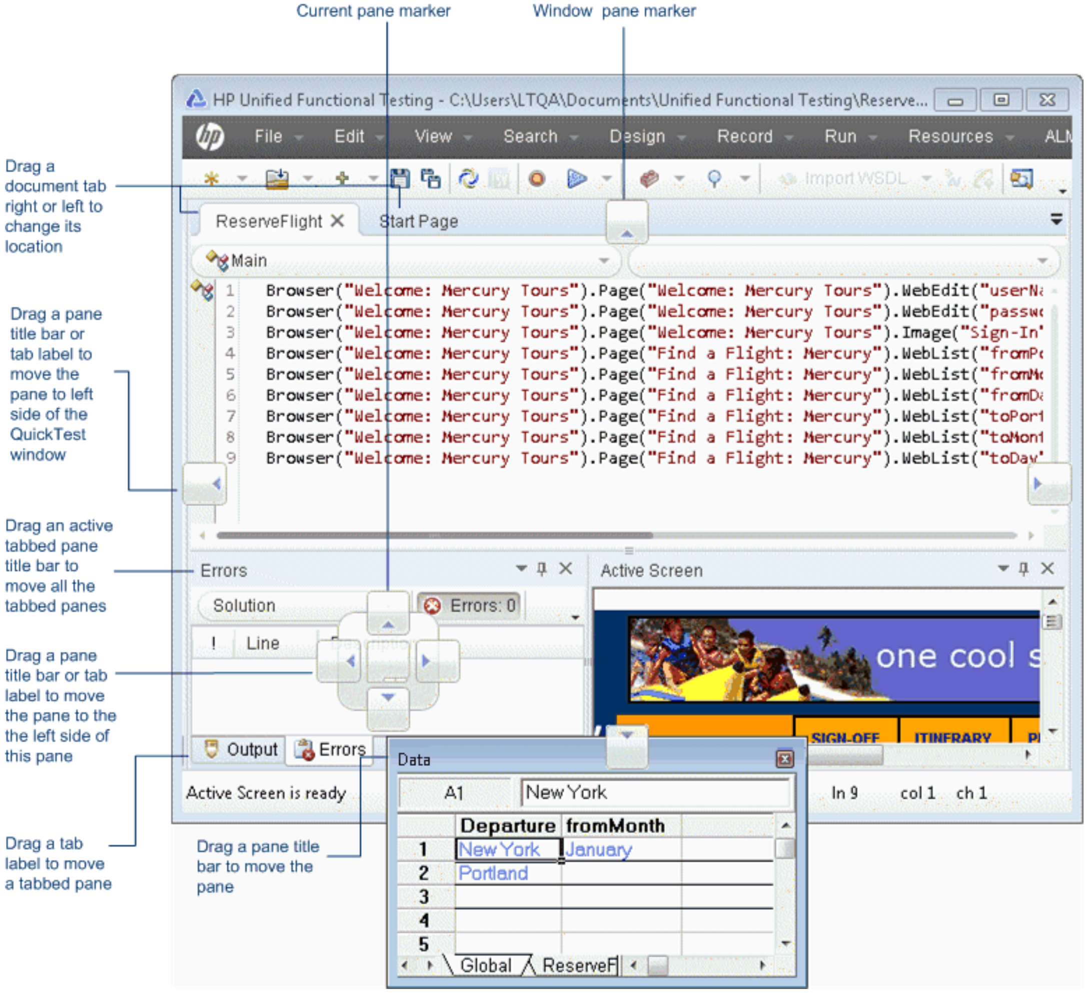
   For example, you can move the Data or Errors pane located at the bottom of the window to be a new tabbed pane at the top right of the window. As you drag the pane, markers are displayed in the active pane and on each edge of the UFT window.

2. Repeat this procedure for each pane you want to move.

During the moving process, the following markers are displayed:

Type	Marker	Description
Current pane markers		Enables you to: ■ position the pane as a new pane in the top, bottom, left or right half, or middle of the active pane, according to the arrow marker selected when you release the mouse button. ■ position the pane as a new tabbed pane in the active window, by releasing the mouse button while selecting the center marker. **Note:** The center marker is displayed only if you are dragging a pane onto an existing pane (other than the document pane).
Window pane markers	▲	Enables you to position the pane across the top of the UFT window.
	▶	Enables you to position the pane across the right side of the UFT window.
	▼	Enables you to position the pane across the bottom of the UFT window.
	◀	Enables you to position the pane across the left side of the UFT window.

The following examples (from left to right) illustrate an example of the UFT window at the beginning of the moving process, during the moving process, and after the moving process, respectively.



Current pane marker

Window pane marker

Drag a document tab right or left to change its location

Drag a pane title bar or tab label to move the pane to left side of the QuickTest window

Drag an active tabbed pane title bar to move all the tabbed panes

Drag a pane title bar or tab label to move the pane to the the left side of this pane

Drag a tab label to move a tabbed pane

Drag a pane title bar to move the pane

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userNa
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("passwo
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("fromPo
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("fromMo
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("fromDa
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("toPort
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("toMont
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("toDay"
```

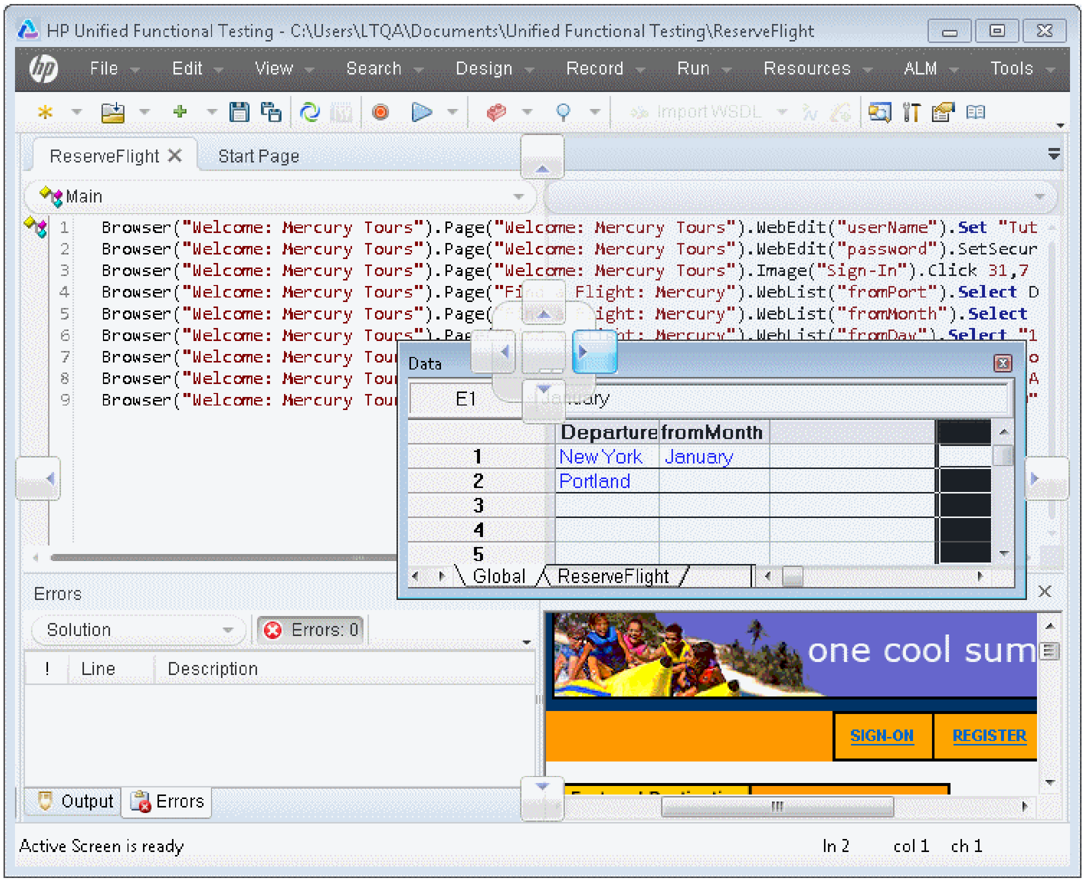
Active Screen is ready

ln 9 col 1 ch 1

A1 New York

	Departure	fromMonth
1	New York	January
2	Portland	
3		
4		
5		

The following illustrates an example of the UFT window during the moving process.



HP Unified Functional Testing - C:\Users\LTQA\Documents\Unified Functional Testing\ReserveFlight
File
Edit
View
Search
Design
Record
Run
Resources
ALM
Tools
Import WSDL
ReserveFlight
Start Page
Main

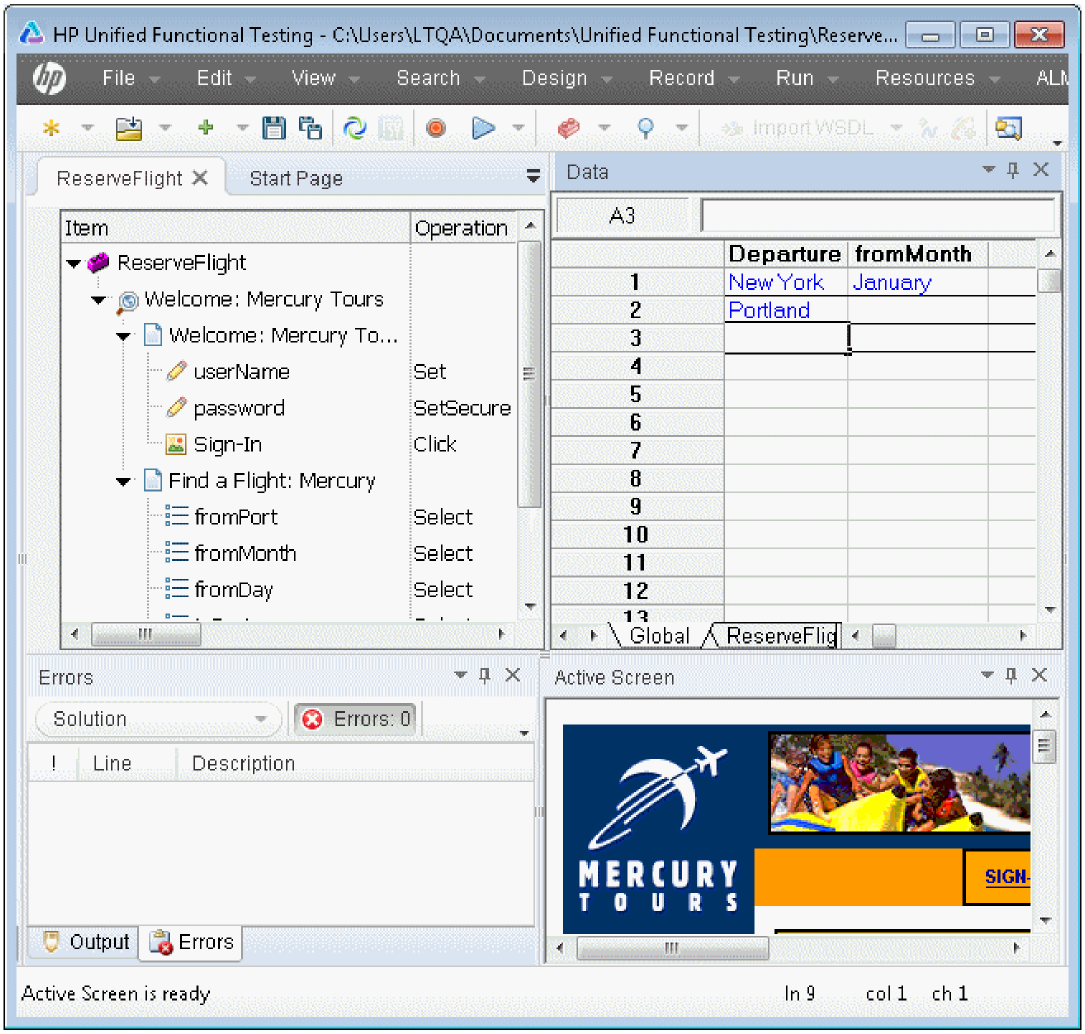
```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "Tut
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetSecur
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click 31,7
Browser("Welcome: Mercury Tours").Page("Fi     Flight: Mercury").WebList("fromPort").Select D
Browser("Welcome: Mercury Tours").Page(       ight: Mercury").WebList("fromMonth").Select
Browser("Welcome: Mercury Tours").Pa               ht: Mercury").WebList("fromDay").Select "1
Browser("Welcome: Mercury Tou
Browser("Welcome: Mercury Tou
Browser("Welcome: Mercury Tou
```

E1 January

	Departure	fromMonth	
1	New York	January	
2	Portland		
3			
4			
5			

Active Screen is ready   ln 2   col 1   ch 1

The following illustrates an example of the UFT window after the moving process.



Item	Operation
ReserveFlight	
Welcome: Mercury Tours	
Welcome: Mercury To...	
userName	Set
password	SetSecure
Sign-In	Click
Find a Flight: Mercury	
fromPort	Select
fromMonth	Select
fromDay	Select

	Departure	fromMonth
1	New York	January
2	Portland	
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		

!	Line	Description

Active Screen is ready ln 9 col 1 ch 1

Show and Hide Panes

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

To auto-hide panes, select the button option in the pane you want to auto-hide and display as a side-tab on one of the edges of the UFT window. The following buttons may be displayed on the title bar:

Button	Description
▾	The **Menu** button enables you to select any of the following: - **Float.** Opens the pane on top of all the other windows and panes as a separate window, with its own title bar. - **Show.** Opens the pane in its default location if it is hidden. - **Dock as Tabbed Document.** Changes the pane to a tab in the document pane. - **Auto-hide.** Displays the pane as a side-tab either at the top or bottom of the UFT window, or on one of the edges, according to its position in the UFT window. - **Hide.** Closes the pane.
📌	The **Auto Hide** button hides the pane. The pane is displayed as a side-tab either at the top or bottom of the UFT window, or on one of the edges, according to its position in the UFT window. To display the pane, hold the cursor over the side-tab. The button toggles to the **Dock** button, shown below.
⊣	The **Dock** button docks the pane to the UFT window. The pane returns the position it was in before it was hidden, and the button toggles to the **Auto Hide** button, shown above.
✕	The **Close** button closes the pane. The pane is removed from the UFT window. To reopen the pane, select it from the **View** menu. **Tip:** You can also close a pane by right-clicking and choosing **Hide** from the context menu.

Float and Dock Panes

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

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

- To float a pane, right-click the title bar, and choose **Float** from the context menu. The pane opens on top of all the other windows and panes, with its own title bar.

- To dock a pane, double-click the title bar, or right-click the title bar and select **Dock as Tabbed Document**. The pane returns to its previous position in the UFT window.

Set the UFT Layout for Record Mode

1. Open a new or existing GUI test or component.
2. Start a recording session.
3. Record one step.
4. Set all of your layout preferences for the recording mode.
5. Stop the recording session.

Set the UFT Layout for Run Mode

1. Enter a breakpoint before the first step in the test. This enables you to arrange the layout during the run session. For details on how to set a breakpoint, see "Breakpoints " on page 681.
2. Run your test or component.
3. When UFT reaches the breakpoint, set all of your layout preferences for the run mode.
4. Stop the run session.

Restore the Default Layout of the UFT Window

You can restore the default UFT window layout for all document types and panes at any time by selecting **View > Reset Window Layout**.

Reference

Tips and Considerations for Customizing the UFT Window Layout

Relevant for: GUI tests and components and API testing

This section includes:

- "Moving Panes " below
- "Showing, Hiding, and Floating Panes " below

Moving Panes

- To move a dockable pane without snapping it into place, press **CTRL** while dragging it to the required location.
- To move a single, tabbed pane, drag the tab label. When you start dragging the tabbed pane, the tab is removed, and its title bar is displayed.
- To move all tabbed panes simultaneously, drag the title bar of the active tabbed pane.

Showing, Hiding, and Floating Panes

- To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and select **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the UFT window, and each pane is displayed only when you hold the cursor over that side-tab.
- If you auto-hide the Errors pane, it is automatically displayed when syntax errors are detected in a test script.
- You cannot auto-hide floating panes or individual tabbed panes.
- By default, auto-hidden panes open across the UFT window, according to their position in the UFT window. For example, if the docked pane was positioned on the right side of the UFT window, it is displayed as a side tab on the right edge of the UFT window, and opens across the right side of the UFT window when selected.
- To float a pane, right-clicking the title bar and choosing **Floating** from the context menu. The pane opens on top of all the other windows and panes, with its own title bar. To dock the pane, double-click the title bar, or right-click the title bar and select **Docking**. The pane returns to its previous position in the UFT window.

Troubleshooting and Limitations - UFT Window Layout

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for the UFT layout on your screen.

When UFT is displayed on a secondary monitor, some drop-down lists may appear on the primary monitor.

Workaround: If you want to ensure that UFT is displayed on only one monitor, display UFT on your primary monitor.

Chapter 5: Active Screen Pane

Relevant for: GUI tests and scripted GUI components

This chapter includes:

Concepts

Active Screen Overview

Relevant for: GUI tests and scripted GUI components

The Active Screen provides a view of your application as it appeared when you performed the corresponding step during a recording or Update Run session.

For Web-based applications the Active Screen uses the actual HTML of the captured page. For Windows-based applications, the Active Screen captures an 'active' snapshot of the page , which includes images of the objects as well as object property data.

An Active Screen can be captured for every step. Additionally, depending on the Active Screen capture options that were set at the time that you recorded or updated your test, the page displayed in the Active Screen can contain detailed property information on each object displayed on the page. For details on setting Active Screen recording options, see "Enhancing Your Tests" on page 815.

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

If UFT captured object information while recording (or updating) your test, you can use the Active Screen to add these objects to the local object repository.

This section includes:

- "Active Screen Capture Settings" below
- "Active Screen Settings for UFT Add-ins" on the next page
- "Active Screen Maintenance" on page 196
- "Tips for Improving Active Screen Performance" on page 196

Active Screen Capture Settings

- You can specify the level at which UFT captures and stores information on objects for Active Screen captures. For example, you can instruct UFT to capture all properties for all test objects on the captured screen, or only the properties of the recorded objects and their parents.
- You can decide how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options. However, more captured information also leads to slower recording and editing times. Removing or decreasing Active Screen information can be especially useful for conserving disk space after you have finished designing the test and you are using the test only for test runs.

- If you find that the information saved in the Active Screen after recording is not sufficient for your test editing needs, or if you no longer need Active Screen information, and you want to decrease the size of your test, you can change the amount of Active Screen information saved with your test.

For details on configuring the Active Screen capture settings, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

Active Screen Settings for UFT Add-ins

Active Screen for Web-based Applications

- The Active Screen displays the captured HTML content using an Internet Explorer browser control, even if you ran your steps on another browser. Additionally, depending on your settings, the Active Screen may capture your HTML page before or after various scripts ran on the page. For these reasons, some pages may look slightly different in the Active Screen than they appeared during your record or update run session, and some property values may be different or may not be available, for example, if those properties are supported one browser, but not in another.

  The Active Screen also stores DOM information in Internet Explorer format. Therefore, if you insert the **Object** property for a Web object when only Active Screen object data is available (the object is not displayed in an open application), then UFT's statement completion feature displays the available native operations and properties for the Internet Explorer DOM, even if you recorded the object in another browser.

  UFT stores the URL path to images and other resources on the page, rather than downloading and storing the images with your test. Therefore, you may need to provide login information to view password-protected resources. For details on accessing password-protected resources in the Active Screen of a Web-based application, see the section on accessing password-protected resources in the active screen in the *HP Unified Functional Testing Add-ins Guide*.

- You can specify Active Screen display criteria for captured Web pages. For example, you can specify whether UFT should load ActiveX controls or Java applets. For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

Active Screen Non-Web-based Applications

- Active Screen pages for non-Web-based applications are based on a single bitmap capture of the visible part of the application window (or other top-level object), with context-sensitive areas representing each object displayed in the Active Screen.

- You can choose whether or not to save the content of the Active Screen with your test. Saving the content of the Active Screen with your test is especially useful if you want to be able to edit the saved test directly from the Active Screen. Later, if you need to conserve disk space after you finish editing the test, and you plan to use your test only for test runs, you can save the test without the content of the Active Screen. (Tests without Active Screen files use significantly less disk space.)

Active Screen Maintenance

As the content of your application changes, you can continue to use the Active Screen from tests that you recorded previously. To do this, you update the selected Active Screen display so that you can use the Active Screen to add new steps to your test rather than re-recording steps on new or modified objects.

> **Example:**
>
> Suppose that one of the pages in your Web site now includes a new object and you want to add a checkpoint that checks this object. You can use the **Change Active Screen** command to replace the page in your Active Screen pane and then proceed to create a checkpoint for this object.

For more details, see "How to Modify the Active Screen Settings" on page 198.

It is also possible to update all Active Screen captures saved with a test using the Update Run Mode. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

Tips for Improving Active Screen Performance

- **For Windows-based applications:** You can choose to save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the Active Screen pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Active Screen** node). The less information saved, the faster your recording times will be. For more details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

- **For Web-based applications:** You can disable the screen capture of all steps in the Active Screen to improve recording time. For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

- If you are testing an application using a UFT add-in, see the *HP Unified Functional Testing Add-ins Guide* to determine whether special Active Screen screen capture options exist for that environment.

- Tests without Active Screen files use significantly less disk space. For more details, see "How to Modify the Active Screen Settings" on page 198.

Tasks

How to Use the Active Screen in Your Test

Relevant for: GUI tests and scripted GUI components

This task describes the different operations you can perform in the Active Screen pane, and includes the following steps:

- "View object information as it appeared during a recording session" below
- "Insert steps on objects in the application" below
- "Insert output values on objects in the application" below
- "Insert checkpoints on objects in the application" below

View object information as it appeared during a recording session

1. Right-click the object in the Active Screen, and select **View/Add Object**. The Object Selection dialog box opens. For details, see "Object Selection Dialog Box " on page 1195.

2. Select the object you want to view and click **OK**. The Object Properties dialog box opens. For details, see "Object Properties Dialog Box" on page 1271.

Insert steps on objects in the application

1. Right-click the object in the Active Screen, and select **Step Generator**. The Object Selection dialog box opens. For details, see "Object Selection Dialog Box " on page 1195.

2. Select the object you want to view and click **OK**. The Step Generator dialog box opens. For details, see "Step Generator Dialog Box" on page 983.

Insert output values on objects in the application

1. Right-click the object in the Active Screen, and select **Insert Output Value**. The Object Selection dialog box opens. For details, see "Object Selection Dialog Box " on page 1195.

2. Select the object you want to view and click **OK**. The Output Value Properties dialog box opens. For details, see "Step Generator Dialog Box" on page 983.

Insert checkpoints on objects in the application

1. Right-click the object in the Active Screen, and select one of the following:

   - **Insert Standard Checkpoint**
   - **Insert Bitmap Checkpoint**

  - **Insert Text Checkpoint**

    The Object Selection dialog box opens. For details, see "Object Selection Dialog Box " on page 1195.

2. Select the object you want to view and click **OK**. The relevant Checkpoint Properties dialog box opens. For details, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

For a user interface description, see "Active Screen Pane User Interface" on page 200.

How to Modify the Active Screen Settings

Relevant for: GUI tests and scripted GUI components

The following steps describe how to modify different settings of the Active Screen:

- "Increase or decrease the Active Screen information saved with a test" below
- "Stop saving Active Screen information and reduce the disk space used by your test" on the next page
- "Update a single Active Screen capture" on the next page
- "Disable Active Screen capture to improve recording time" on the next page

Increase or decrease the Active Screen information saved with a test

1. Modify the Active Screen capture preference in the Active Screen pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Active Screen** node) to capture the amount of information you need. For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

2. Perform one of the following:

   - **Update Run Mode.**This operation enables you to save the required amount of information in the Active Screen for all existing steps. For details on the **Update Run Mode** options, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

   - **Re-record the steps**. This enables you to specify the objects you want to add to the Active Screen by performing one of the following:

     - Select the step after which you want to record your step, position your application to match the selected location in your test, and then begin recording.

     - Place a breakpoint in your test at the step before which you want to add a step and run your test to the breakpoint. This brings your application to the point from which to record the step. For details on setting breakpoints, see "Breakpoints " on page 681.

Stop saving Active Screen information and reduce the disk space used by your test

1. Open the relevant test.

2. Select **File > Save As** and clear the **Save Active Screen files** check box.

   > **Note:** If you clear this check box, your Active Screen files will not be saved, and you will not be able to edit your test using the options that are normally available from the Active Screen.

3. Click **Save** to apply your changes. For details, see "Save <Resource>/Save <Document> As Dialog Box" on page 164.

Update a single Active Screen capture

1. Make sure that your application is displaying the window or page that you want to use to replace what is currently displayed in the Active Screen pane.

2. In the Keyword View, click a step that you want to change. The window or page is displayed in the Active Screen pane.

3. Select **Tools > Change Active Screen**. The UFT window is hidden and the mouse pointer becomes a pointing hand. For details on using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 1196.

4. Click the window or page displayed in your application.

5. When a message prompts you to change your current Active Screen display, click **Yes**.

Disable Active Screen capture to improve recording time

Click **Custom Level** to open "Custom Active Screen Capture Settings Dialog Box" on page 551, and select the **Disable Active Screen Capture** option.

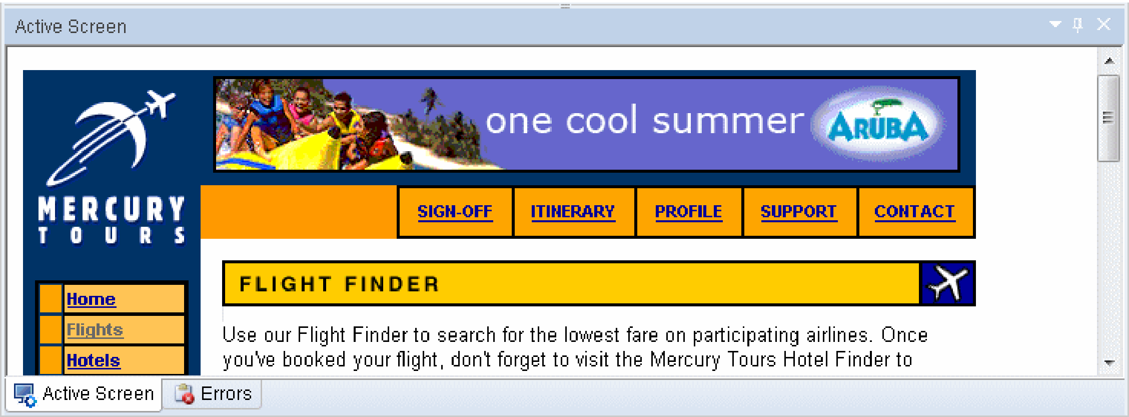
For more details, see "Tips for Improving Active Screen Performance" on page 196.

Reference

Active Screen Pane User Interface

Relevant for: GUI tests and scripted GUI components

This pane enables you to view snapshots of your application as it appeared during a step in a recording session. It also enables you to parameterize object values and insert steps, checkpoints, methods, and output values for almost any object in the page at any time after the recording session.



Use our Flight Finder to search for the lowest fare on participating airlines. Once you've booked your flight, don't forget to visit the Mercury Tours Hotel Finder to

To access	1. 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. 2. Select **View > Active Screen**.
Important information	• You can increase or decrease the Active Screen information saved with your test. For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547. • Additional context menu options may be available depending on the test object selected in the Active Screen.
Relevant tasks	• "How to Modify the Active Screen Settings" on page 198 • "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199
See also	• "Active Screen Overview" on page 194 • "Custom Active Screen Capture Settings Dialog Box" on page 551

User interface elements and context menu options are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<active screen area>	The snapshot of the application at the time of recording the selected step. The object used in the step is highlighted.
Insert Standard Checkpoint	Inserts a standard checkpoint on the selected object. For details, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.
Insert Output Value	Inserts an output value on the selected object. For details, see "Output Values Overview" on page 1488.
View / Add Object	Opens the Object Properties dialog box, which enables you to view object properties and add the selected object to your local object repository. For details, see "Object Properties Dialog Box" on page 1271.
Step Generator	Opens the Step Generator dialog box, which enables you to create a step using the selected object. For details, see "Step Generator Dialog Box" on page 983.
Insert Bitmap Checkpoint	Inserts a bitmap checkpoint on the selected object. For details, see the areas relevant to bitmap checkpoints in the "Checkpoint Properties Dialog Box" on page 1432
Insert Text Checkpoint	Inserts a text checkpoint on the selected object. For details, see the areas relevant to text/text area checkpoints in the "Checkpoint Properties Dialog Box" on page 1432.

Troubleshooting and Limitations - Active Screen Pane

Relevant for: GUI tests and components

- For steps that contain Insight test objects, the Active Screen provides only a visual reference to the application and the test object's context. The Active Screen displays the Insight test object highlighted within a screen capture of its parent object.

  You cannot use the Active Screen of an Insight step to view object properties, to insert checkpoints or output values, or to add objects to the object repository.

- The OCR text recognition mechanism is not supported for objects in the Active Screen.

- When creating objects, checkpoints, or output values from the Active Screen for objects in a modal dialog box within a Web browser, they are created under the Browser object and not under the Window object.

  Workaround: Add the object directly from the application, for example by using the **Add Objects to Local** button in the Object Repository window.

Chapter 6: Bookmarks Pane

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This chapter includes:

Concepts

Bookmarks Overview

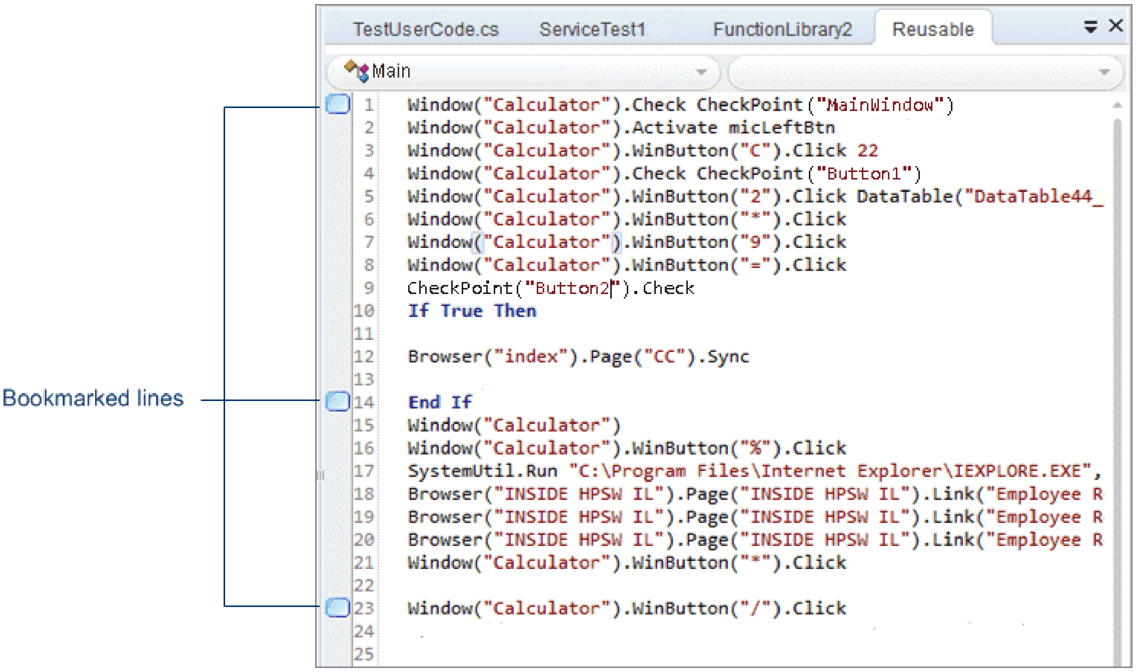
Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

You can use bookmarks when editing actions, scripted components, function libraries, or user code files in the Editor. Bookmarks enable you to navigate between sections of your document more easily. The bookmarks are saved on your file system on a per-user basis.

When you assign a bookmark, an icon is added to the left of the selected line of your document.

GUI testing example

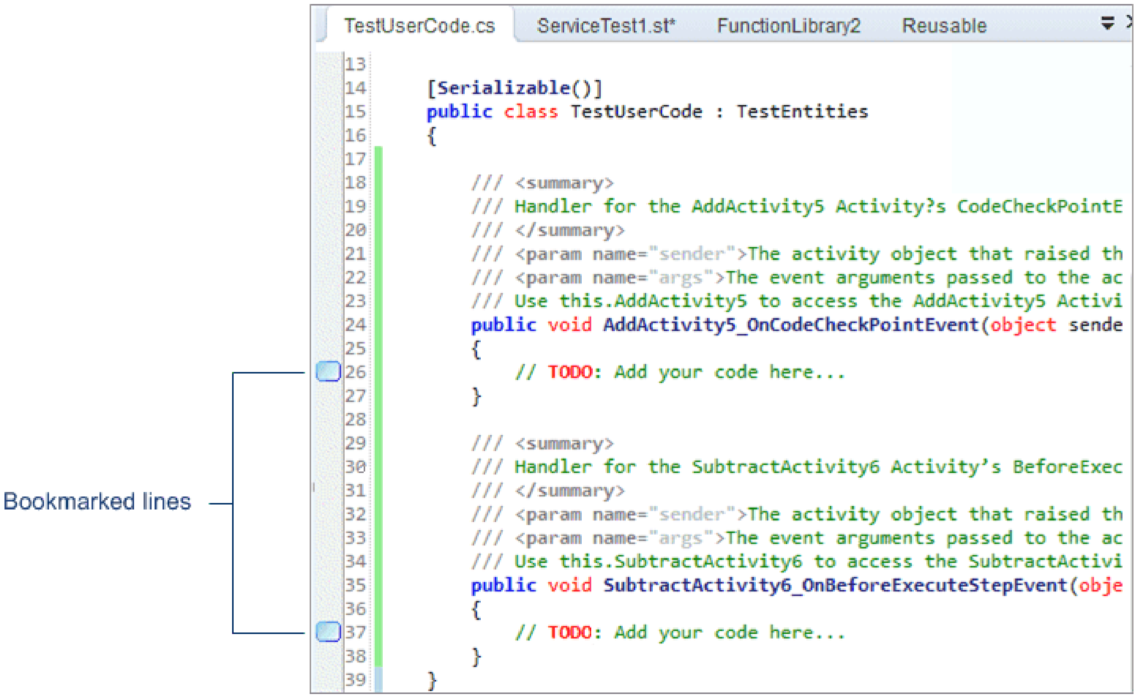
In the following example, bookmarks have been added to the reusable GUI action in the Editor:



```
Window("Calculator").Check CheckPoint("MainWindow")
Window("Calculator").Activate micLeftBtn
Window("Calculator").WinButton("C").Click 22
Window("Calculator").Check CheckPoint("Button1")
Window("Calculator").WinButton("2").Click DataTable("DataTable44_
Window("Calculator").WinButton("*").Click
Window("Calculator").WinButton("9").Click
Window("Calculator").WinButton("=").Click
CheckPoint("Button2").Check
If True Then

Browser("index").Page("CC").Sync

End If
Window("Calculator")
Window("Calculator").WinButton("%").Click
SystemUtil.Run "C:\Program Files\Internet Explorer\IEXPLORE.EXE",
Browser("INSIDE HPSW IL").Page("INSIDE HPSW IL").Link("Employee R
Browser("INSIDE HPSW IL").Page("INSIDE HPSW IL").Link("Employee R
Browser("INSIDE HPSW IL").Page("INSIDE HPSW IL").Link("Employee R
Window("Calculator").WinButton("*").Click

Window("Calculator").WinButton("/").Click
```

API testing example

In the following example, bookmarks have been added to anAPI event handler:



```

[Serializable()]
public class TestUserCode : TestEntities
{

    /// <summary>
    /// Handler for the AddActivity5 Activity?s CodeCheckPointE
    /// </summary>
    /// <param name="sender">The activity object that raised th
    /// <param name="args">The event arguments passed to the ac
    /// Use this.AddActivity5 to access the AddActivity5 Activi
    public void AddActivity5_OnCodeCheckPointEvent(object sende
    {
        // TODO: Add your code here...
    }

    /// <summary>
    /// Handler for the SubtractActivity6 Activity’s BeforeExec
    /// </summary>
    /// <param name="sender">The activity object that raised th
    /// <param name="args">The event arguments passed to the ac
    /// Use this.SubtractActivity6 to access the SubtractActivi
    public void SubtractActivity6_OnBeforeExecuteStepEvent(obje
    {
        // TODO: Add your code here...
    }
}
```

The Bookmarks pane displays a list of all bookmarks added in any document connected with the open test. Using the Bookmarks pane, you can:

- Add a new bookmark.
- Navigate to the location of the bookmark in your document.
- Navigate between consecutive bookmarks in the pane.
- Delete an individual bookmark.
- Clear all bookmarks.

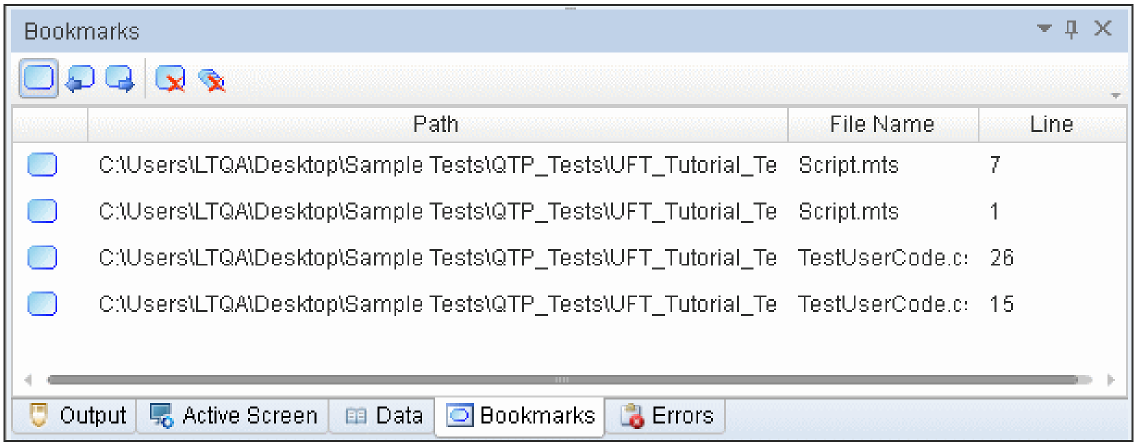
For task details, see "How to Use Bookmarks in the Editor" on page 321.

Reference

Bookmarks Pane User Interface

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

The Bookmarks pane displays the location of bookmarks in your action, scripted component, function library, or user code files and enables you to navigate to these bookmarks.



	Path	File Name	Line
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	Script.mts	7
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	Script.mts	1
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	TestUserCode.c:	26
	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UFT_Tutorial_Te	TestUserCode.c:	15

To access	Select **View > Bookmarks**.
Important information	• All bookmarks added to test actions, scripted components, function libraries, or user code file included in a saved solution are maintained after you close and reopen the pane, document, or UFT. • Bookmarks are saved per solution, even if multiple users are working on the same documents. • When working with documents saved in ALM, bookmarks are maintained until you close UFT.
Relevant tasks	"How to Use Bookmarks in the Editor" on page 321
See also	"Bookmarks Overview" on page 204

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

UI Element	Description
<bookmarks list>	All bookmarks connected with the open solution are displayed in the Bookmarks pane. Even if a test is not open in the document pane, its bookmarks are still displayed in the Bookmarks pane. Bookmarks are removed from the pane when the solution is closed. **Example:** When you open a solution, bookmarks in any test's actions and associated function libraries (for GUI tests) or any relevant user code files (for API tests) are displayed. For each bookmark in the list, UFT displays the location of the bookmark's document, the file name, and the line location of the bookmark. For documents stored on ALM, the location of the local temporary copy is displayed. Bookmarks inserted in a document or user code document are kept with the document when it is saved in an ALM project. **Note:** You can double-click the bookmark line to navigate directly to the relevant line in your document. If a bookmark's document is not open, UFT opens the relevant document.
	Insert/Remove Bookmark. Inserts a bookmark in the current document in the line where the cursor is currently located.
	Previous Bookmark. Navigates to previous bookmark in the pane.
	Next Bookmark. Navigates to next bookmark in the pane.
	Delete. Deletes the currently selected bookmark from the relevant document and from the list displayed in the pane.
	Clear All Bookmarks. Deletes all bookmarks you have added from all documents and from the list displayed in the pane.

Chapter 7: The Canvas

Relevant for: GUI tests and API testing

This chapter includes:

Concepts

The Canvas

Relevant for: GUI tests and API testing

The canvas provides a visual representation of the GUI or API test flow. It opens as a tab in the document pane.

For a GUI test, the canvas shows the flow of your test actions. For an API test, the canvas shows the steps included in your test.

Business process tests and flows are displayed only in a grid view, and are not shown in the canvas at all. For details about viewing business process tests and flows in the document pane, see "Business Process Testing in UFT User Interface" on page 2074. For details about application areas, see "Application Area User Interface" on page 2101.

The canvas has the following features:

Test Action Management (GUI Testing only)

You can access the Action Properties, as well as the Action Call Properties from the action on the canvas. Using these properties, you can set the settings and properties for how your action is used in the test.

For details on the Action Properties, see "Action Properties Dialog Box" on page 900. For details on the Action Call Properties, see "Action Call Properties Dialog Box" on page 895.

Test Runs from a Specific Action (GUI Testing only)

Using the action context menu, you can run the test from a specific action or debug the test from a specific action. As a result, you do not need to run an entire test to check if the selected action runs as expected.

For details on using these commands, see "Running to a Step and Debugging from a Step " on page 678.

Step Manipulation

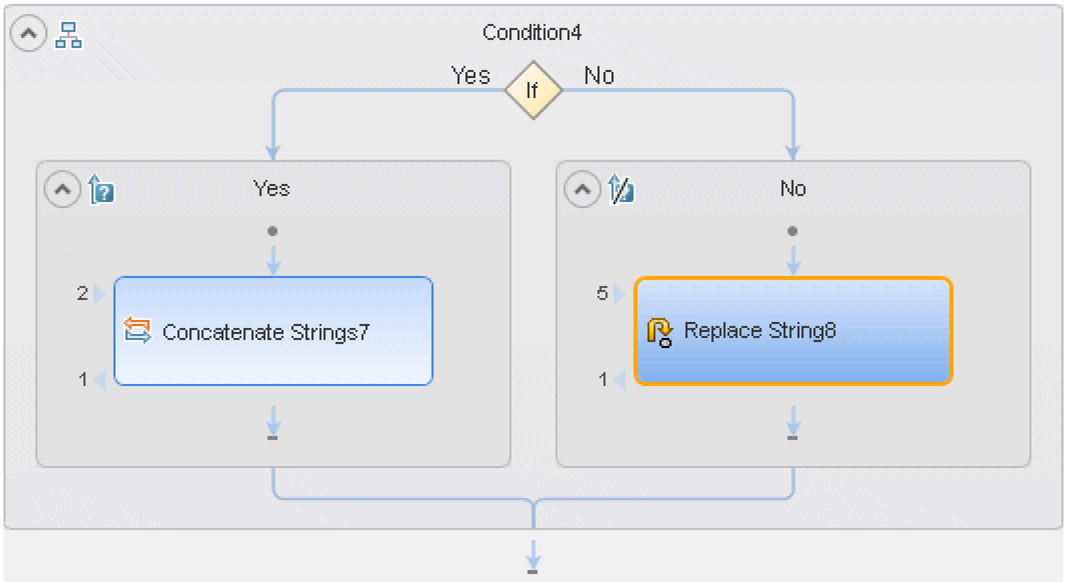
The canvas enables you to manipulate the steps in the following ways:

- **Reorder**. Drag a step (for API tests) or an action (for GUI tests) from one location to another.
- **Move Up or Move Down. (GUI Tests only)** You can change the order of actions using the Move Up or Move Down commands from the action context menu.
- **Copy and Paste. (API Tests only)** Select a step and press CTRL+C to copy it to the clipboard. Press CTRL+V to paste it into another location within the Test Flow or to another loop.
- **Delete. (API Tests only)** Select a step and press the keyboard's **Delete** button.

Flow Control (API Testing only)

By default, the canvas includes a Test Flow area to serve as a container for test steps. However, you can also add special test flow steps including:

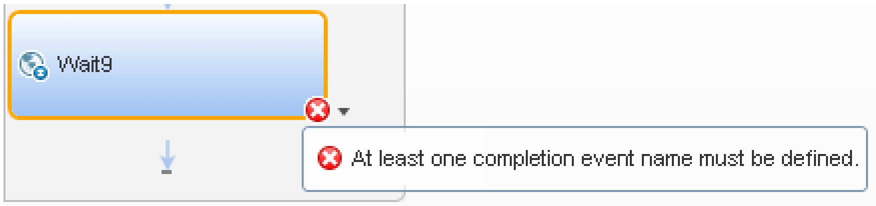
- **Loop.** The main **Test Flow** area provides a default loop for test steps. However, you can add additional loops to the test. Using the Properties pane, you can specify the type of loop, number of iterations, and conditions for the Test Flow or loop.
- **Conditional steps.** Adds a two-branched conditional node to the test. You then add the activities as part of the conditional branches.



- **Break.** Moves the test run to the step after the loop.
- **Continue.** Moves the test run to the first step in the surrounding **Test Flow** or **Loop** and begins the next test iteration.

Alerts (API Testing only)

Steps that require additional information in order to run, such as **HTTP Request/Response** or **SOAP Request**, **Custom Code**, and **Wait** post an alert in their top right corner. Clicking an alert button displays the missing values for the step. If you click on the alert details, the Properties pane opens to the required property to resolve the alert.

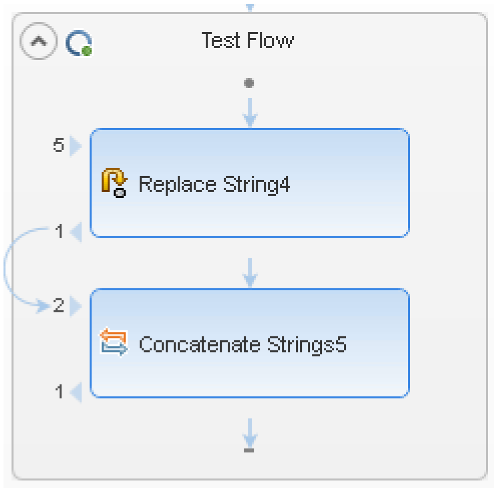


At least one completion event name must be defined.

You can also view this information in the Errors pane by selecting **View > Errors**. For details about the Errors pane, see "Errors Pane" on page 364.

Step Input/Output Property Connections (API Testing only)

The canvas displays the relationship between output and input properties/parameters. An arrow indicates the relationship between the linked properties/parameters.

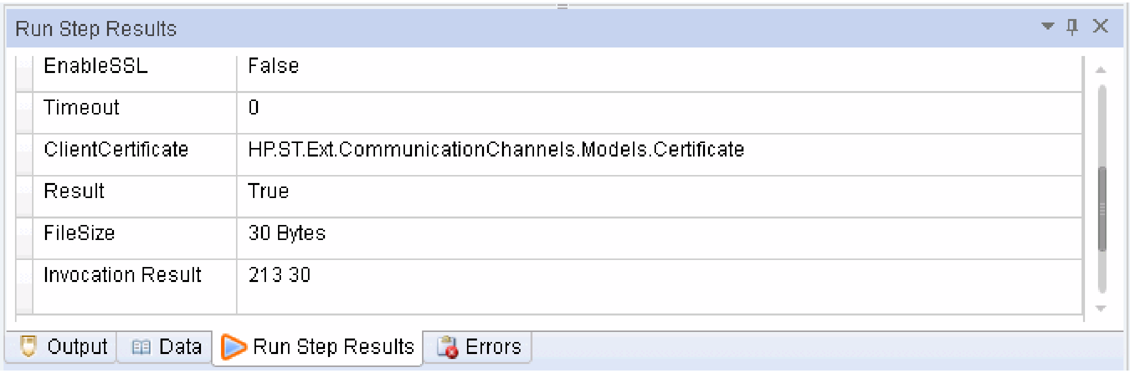


For details, see the "Select Link Source Dialog Box (API Testing)" on page 1840.

For details about linking multiple steps to a single result, see "Outgoing Links" on page 1808.

Tests of Individual Steps (API Testing only)

The **Run Step** utility lets you run a single test step without running the entire test. Using the property values from the Properties pane, the **Run Step** operation invokes the step locally and presents the results in the **Run Step Results** pane. The following example shows the results of an **FTP File Get Size** step.



EnableSSL	False
Timeout	0
ClientCertificate	HP.ST.Ext.CommunicationChannels.Models.Certificate
Result	True
FileSize	30 Bytes
Invocation Result	213 30

For more details about individual step runs, see the "Run Step Results Pane" on page 454.

For details about the **Run Step** utility, see the "Run Step Dialog Box" on page 459.

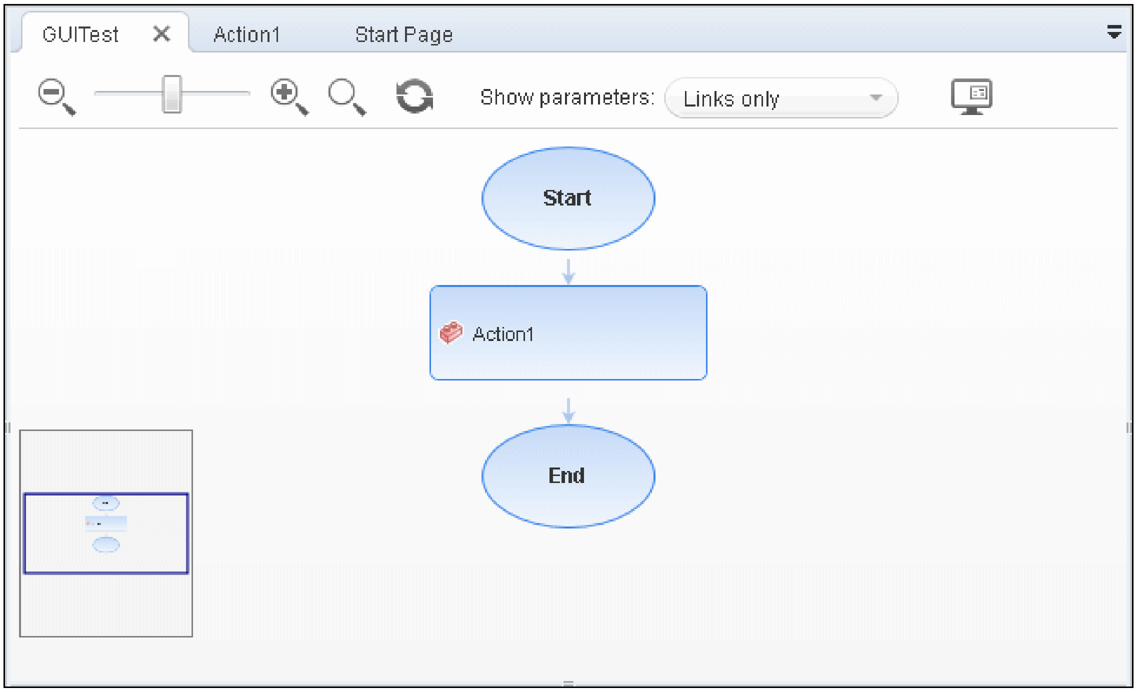
Reference

Canvas User Interface (GUI Testing)

Relevant for: GUI tests and components

The canvas provides a visual representation of the actions in the test flow and enables you to:

- Manage actions and change their order in the test
- Run your test from a selected action or to a selected action
- Debug your test from a selected action
- Display actions in the Keyword View or the Editor



To access	1. Create or open a GUI test. 2. Click the test tab.

Important information	• The canvas is displayed by default when you create a new test. • You can close the test canvas by right-clicking a canvas tab and selecting **Close**. Action tabs included in the test canvas remain open. • If you close a test canvas tab, the next time you open a test, the canvas tab will not open. To open the canvas tab, do one of the following: ▪ Double-click the **<test name>** node in the Solution Explorer ▪ Select **View > Reset Window Layout**. • Working with the canvas closed can improve UFT's performance when creating or opening a GUI test. • For details on working with actions in your test, see "Actions in GUI Testing" on page 871.
Relevant tasks	• "How to Display and Modify Action-Related Information" on page 886 • "How to Manage the Structure of Your GUI Test" on page 888
See also	"The Canvas" on page 209

This pane includes the following icons and context menu options:

- "Main User Interface Elements" on the next page
- "Action Type Icons" on page 215
- "Context Menu Options - Test Nodes" on page 215
- "Context Menu Options - Action Nodes" on page 215
- "Context Menu Options - API Test Nodes" on page 216

Main User Interface Elements

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

UI Element	Description
<Zoom slider>	Enables you to zoom in or out of the test flow. Do one of the following: • Click the magnifying glass with the plus sign to zoom in on (enlarge) the test flow. • Click the magnifying glass with the minus sign to zoom out of (shrink) the test flow. • Click and drag the zoom slider position handle left or right to zoom in or out of the test flow.
🔍	**<Return to default size button>.** Restores the test flow to the default size and moves the zoom slider position to the center.
Show parameters	Enables you to select the types of parameter information shown in the test flow, including: • **None.** Shows only the number of input and output parameters adjacent to the relevant action. Click the number to expand the parameter details. • **Links only.** Shows links when parameters are defined as input from other actions. • **Links and names.** Shows links and parameter names, when parameters are defined as input from other actions. • **All.** Displays the names and types of each input and output parameter adjacent to the relevant action. Click the number to collapse the parameter details. • **User defined.** Shows only user-defined parameters and hides built-in parameters.
🖥	**<Minimap display button>.** Located above the test flow on the right. Displays or hides the test flow minimap in the lower left corner of the canvas. Click to show or hide the minimap. The minimap display button is highlighted in grey when the minimap is displayed, and is not highlighted when the minimap is hidden.

UI Element	Description
<test flow>	Displays the actions in the test in a flow chart format, as well as any parameter details available, as specified in the **Parameter presentation** drop-down list. For details, see "Action Type Icons" below and "Context Menu Options - Test Nodes" below.
<Minimap>	A small, high-level view of the entire test flow, useful when using large flows and/or small screens. **Note:** When you zoom out of the canvas, the test flow in the minimap becomes smaller because the minimap displays the entire test flow.

Action Type Icons

Icon	Description
🧱	A call to a reusable or non-reusable action.
↻	A call to an API test or action.

Context Menu Options - Test Nodes

Test node context menus are accessible by right-clicking the **Start** and **End** nodes in the test flow.

Command	Description
Settings	Opens the "Properties Pane (Test/Business Component Settings Dialog Box)" on page 590, which enables you to view the test settings.

Context Menu Options - Action Nodes

Action node context menu items are accessible by right-clicking an action in the test flow.

Command	Description
Action Call Properties	Opens the "Action Call Properties Dialog Box" (described on page 895), which enables you to view details of the selected action call.
Action Properties	Opens the "Action Properties Dialog Box" (described on page 900), which enables you to view details of the selected action.
Object Repository	Opens the "Object Repository Window" (described on page 1274), which displays a tree containing all objects in the current test.
Call to New Action	Opens the "Insert Call to New Action Dialog Box (GUI Testing)" (described on page 913), which enables you to insert a call to a new action.

Command	Description
Call to Copy of Action	Opens the "Select Action Dialog Box " (described on page 915), which enables you to select a copy of an action to insert into the test.
Call to Existing Action	Opens the "Select Action Dialog Box " (described on page 915), which enables you to insert a call to a specific existing reusable action.
Delete	Removes the selected action from your test. For details, see "How to Manage the Structure of Your GUI Test" on page 888. **Note:** This option is disabled if your test contains only one action.
Run from Action	Starts a run session from the beginning of the selected action.
Debug from Action	Starts a debug session and pauses it at the beginning of the selected action.
Run to Action	Runs the test until the beginning of the selected action and then pauses the run session.
Move Up	Moves a action (a direct child of the test) up the canvas tree. **Note:** This option is disabled if you only have one action in your test.
Move Down	Moves a top-level action (a direct child of the test) down the canvas tree. **Note:** This option is disabled if you only have one action in your test.

Context Menu Options - API Test Nodes

API test node context menu items are accessible by right-clicking a call to an API test within a GUI test action.

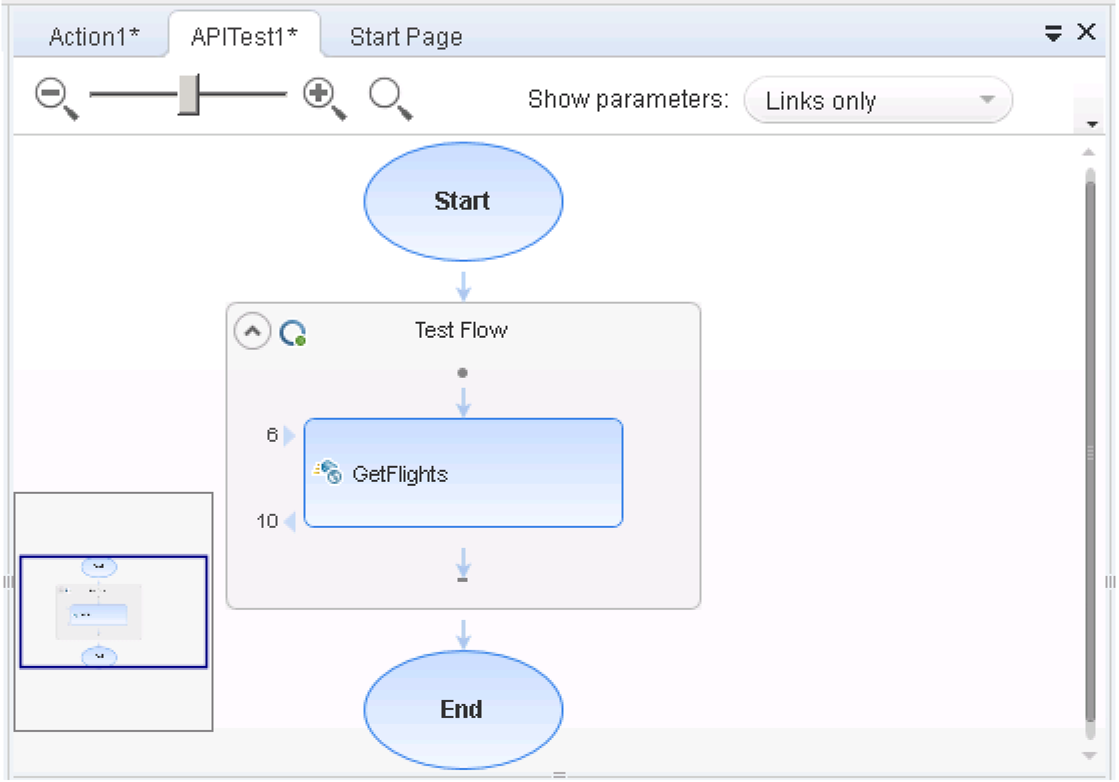
Command	Description
Edit Call to API Test	Opens the "Call to API Test/Action Dialog Box" on page 1075 (described on page 1075), enabling you to view and modify your API test parameters. **Note:** To replace the current call with a different API test, you must first delete the current call and then insert a new one.
Delete	Removes the selected API test call from your test.

Canvas User Interface (API Testing)

Relevant for: API testing only

The canvas is displayed by default when you create a new test. It provides a visual representation of the test flow and enables you to:

- Display test and action steps
- Manage steps and change their order
- Test individual steps using the Run Step command
- Open wizards and dialog boxes to resolve steps with missing information.



To access	1. Create or open an API test or component. 2. Click on the test tab.
See also	"Activity Overview" on page 1612

This pane includes the following icons and context menu options:

- "Main User Interface Elements" below
- "Context Menu Options" on the next page

Main User Interface Elements

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

UI Element	Description
<Zoom slider>	Enables you to zoom in or out of the test flow image. Do one of the following: • Click the magnifying glass with the plus sign ⊕ to zoom in on (enlarge) the test flow image. • Click the magnifying glass with the minus sign ⊖ to zoom out of (shrink) the test flow image. • Click and drag the zoom slider position handle left or right to zoom in or out of the test flow image.
🔍	**<Return to default size>.** Returns the test flow image to the default size and the zoom slider position to the center.
Show parameters	Enables you to select the level of information shown in the canvas, including: • **None.** Shows only the number of input and output properties, adjacent to each step or action frame. Clicking the number shows the property names. • **All.** Displays the names of all the input and output properties and their links, with the total number of properties. • **Links only.** Displays only the links to other steps in the flow, and the number of properties. (API tests only) **Tip:** Clicking the number displays the properties for the step. • **Links and names.** Displays the links to other steps, the name of the linked properties, and the number of properties. (API tests only) • **Custom.** Indicates that the **Show parameters** settings for some of the steps were modified manually and not via the drop down. This entry is not selectable.

UI Element	Description
🖥	**<Minimap display>.** Located above the test flow image on the right, and displays or hides the test flow minimap in the lower left corner of the canvas. The toggle button is highlighted when the minimap is displayed.
<Test flow>	Displays the test flow in a flow chart format, as well as any parameter details available, as specified in the **Show parameters** drop-down list.
<Minimap>	A small, high-level view of the entire test flow, useful when using large flows and/or small screens. **Note:** When you zoom out of the canvas, the area displayed in the canvas becomes larger. This has the effect of shrinking the test flow image in the minimap because the minimap displays the entire test flow.

Context Menu Options

Context menu items are accessible by right-clicking a step in a test or action.

Option	Description
Cut	Cuts the selected step and places it on the clipboard.
Copy	Copies the selected step to the clipboard.
Delete	Deletes the selected step.
Properties	Opens the Properties Pane. For details, see "Properties Pane" on page 385.
Open	Opens the selected action step in its own tab (for action steps inside the Test Flow).
Paste	Pastes the step currently on the clipboard to the location of the cursor. **Note:** • Available only after you add a step to the clipboard using either **Cut** or **Copy**. • Only visible when clicking on the down arrow inside the Test Flow, Loop, or Action frame.

Option	Description
Run Step	Runs the current step using its properties from the Properties pane, and shows the results in the **Run Step Results** pane. For details, see "Run Step Results Pane" on page 454. **Note:** • This command is not available for all steps. • If an input property was linked to an output property of a previous step, the Run Step dialog box opens. This lets you set constant values for the step's properties. For details, see "Run Step Dialog Box" on page 459.

Chapter 8: Data Pane

Relevant for: GUI testing, API testing, and business process testing

This chapter includes:

Concepts

Data Pane Overview

Relevant for: GUI testing, API testing, and business process testing.

UFT enables you to insert and run steps that are driven by data in the Data pane.

This section includes:

- "For GUI testing" below
- "For API testing" below
- "For business process testing" on the next page

For GUI testing

The data your test uses is stored in a **design-time** data table, which is displayed in the Data pane while you insert and edit steps.

The Data pane has the characteristics of a Microsoft Excel spreadsheet, enabling you to store and use data in its cells and also perform mathematical formulas within the cells.

You can use the GUI testing Data pane, or you can use any Microsoft Excel (.xls or .xlsx) file.

You can use the **DataTable**, **DTSheet** and **DTParameter** utility objects to manipulate the data in any cell in the Data pane. For details on these objects, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

You can insert data table parameters and output values into your test. Using data table parameters and/or output values in a test enables you to create a **data-driven** test or action that runs several times using the data you supply.

For more details, see "How UFT Uses Data Tables in GUI Testing" on the next page.

For API testing

You can designate an Excel spreadsheet, a local data table, an XML file, or results of a database query as data to be used for your test.

Using the Properties pane, you can insert data in your test step's input and output values. Using these parameters enables you to create a **data-driven** test.

> **Tip:** If UFT is integrating with ALM, you can use the data awareness features to:
>
> - use a different data resource for each run iteration
> - use the same data resource in multiple tests
> - perform other tasks
>
> For details, see "Data Awareness in ALM" on page 715 and "How to Data Drive a Test Using Data from ALM" on page 720.

For details about data handling for API tests, see "Data Usage in API Tests" on page 1800.

For business process testing

The Data pane enables you to view and modify iteration data for a business component or group in your test or flow, or for a specific flow in your test. You can add and delete iterations, populate parameter values, link input parameter values to other parameters in your test or flow, and promote component or flow parameters to the next level.

You can configure components, flows, and groups to run a specified number of iterations during a single run. Each iteration can use different parameter values.

For each iteration, use the Data pane to configure with which data a component, flow, or group runs in a test or flow.

For more details, see "Considerations for BPT Test Iterations and Parameters" on the next page.

> **Note:** For task details on defining iterations for components, groups, and flows, see "Iterate components and flows" on page 2072.

How UFT Uses Data Tables in GUI Testing

Relevant for: GUI tests and components

In each repetition, or **iteration**, of a run session, UFT uses a different value from the data table.

During the run session, UFT creates a **run-time** data table—a live version of the data table associated with your test. During the run session, UFT displays the run-time data in the Data pane so that you can see any changes to the data table as they occur.

When the run session ends, the run-time data table closes, and the Data pane again displays the stored design-time data table. Data entered in the run-time data table during the run session is not saved with the test. The final data from the run-time data table is displayed in the **Run-Time Data Table** in the Run Results Viewer. For details on the run-time data table, see the section on the Run Results Viewer Data Pane (described in the *HP Run Results Viewer User Guide*).

> **Tip:** If UFT is integrating with ALM, you can use the data awareness feature to:
>
> - use a different data table for each run iteration
> - use the same data table in multiple tests
> - perform other tasks.
>
> To use this feature, make sure to save your data table parameters in the **Global** sheet.
>
> For details, see "How to Data Drive a Test Using Data from ALM" on page 720.

Parent Topic: "Data Pane Overview" on page 222

Considerations for BPT Test Iterations and Parameters

When working with iterations and parameters for business process test, consider the following:

- "Match an application’s post-condition with the next iteration’s pre-condition" below
- "Link input parameter values to output parameters" below
- "Moving a group, or a member within a group, can cause a parameter reference to be removed" on the next page

Match an application’s post-condition with the next iteration’s pre-condition

For a business component to run iterations successfully, it is essential that the post-condition (the state of the application after the last step in the component runs) match the pre-condition (the state of the application before the first step in the component runs).

For group iterations to be successful, the state of the application at the end of the last item in the group must match the state of the application before the first item in the group.

For example, if the first component in the group assumes that the Login dialog box in an application is open, then at the point where the last component of the group ends, the Login dialog box must be in an open state before the next iteration begins.

> **Note:** Components or flows in a group with input parameters must have the same number of iterations.

Link input parameter values to output parameters

Iterations in a business process test or flow can result in multiple output parameter values. By linking parameters, each iteration can pass its output value as input to the corresponding target component or flow.

When you use the output of a previous component as the value for an input component parameter, the option applies to all component iterations for that input parameter.

To link input parameters values to other test or component parameters, use the "Select Link Source Dialog Box (BPT)" on page 2128 (described on page 2128).

Moving a group, or a member within a group, can cause a parameter reference to be removed

When a group is moved to a position preceding the component that provides an input component parameter needed by a parameter in the group, the parameter linkage is deleted, and the value for the source parameter will be empty.

You can then either supply a value for the parameter or reinstate the link using the "Select Link Source Dialog Box (BPT)" on page 2128.

Data Table Sheets for GUI Testing

Relevant for: GUI tests and scripted GUI components

When working with tests, the data table has two types of data sheets—**Global** and **Action**. You can access the different sheets by clicking the appropriate tabs below the data table.

- **"Global Sheet".** You store data in the Global tab when you want it to be available to all actions in your test and you want the data to control the number of test iterations.

  You must also store data in the Global tab if your test is stored in ALM, and you want to use the data awareness feature. For details, see "Data Awareness in ALM" on page 715.

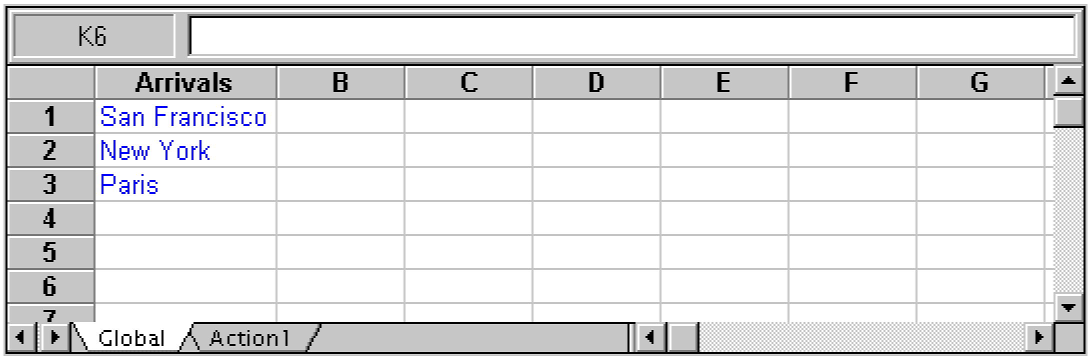
- **"Action Sheets".** You store data in the action's tab when you want to use the data in data table parameters for that action only and you want the data to control the number of action iterations.

For example, suppose you are creating a test on the sample Mercury Tours Web site. You might create one action for logging in, another for booking flights, and a third for logging out. You may want to create a test in which the user logs onto the site once, and then books flights for five passengers. The data about the passengers is relevant only to the second action, so it should be stored in the action sheet corresponding to that action.

Global Sheet

Relevant for: GUI tests and scripted GUI components

The Global sheet contains the data that replaces parameters in each iteration of the test. If you create a Global parameter called Arrivals, the Global sheet might look like this:



K6

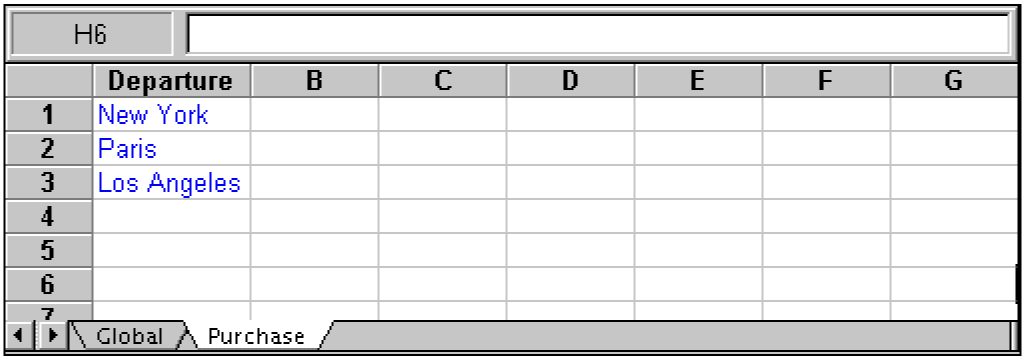
	Arrivals	B	C	D	E	F	G
1	San Francisco						
2	New York						
3	Paris						
4							
5							
6							
7							

For details on creating global parameters, see "Parameterizing Object Values" on page 1526.

Action Sheets

Relevant for: GUI tests and scripted GUI components

Each time you add a new action to the test, a new **action sheet** is added to the Data pane. Action sheets are automatically labeled with the exact name of the corresponding action. The data contained in an action sheet is relevant for data table parameters in the corresponding action only. For example, if a test had the data table below, your GUI test would use the data contained in the Purchase sheet when running iterations on action parameter steps within the **Purchase** action.



H6

	Departure	B	C	D	E	F	G
1	New York						
2	Paris						
3	Los Angeles						
4							
5							
6							
7							

For details on creating action parameters, see "Parameterizing Object Values" on page 1526.

> **Note:** If UFT is integrating with ALM, make sure to save your data table parameters in the "Global Sheet" (described on page 226). For details, see "Data Awareness in ALM" on page 715.

Data Pane - Save Options for GUI Testing

Relevant for: GUI tests and scripted GUI components

The Data pane contains the values that UFT substitutes for data table parameters when you run a test, as well as any other values or formulas you enter. Whenever you save your test, UFT automatically saves its data table as an .xls or .xlsx file.

When working with tests, the data table is saved with your test by default. You can save the data table in another location and instruct the test to use this data table when running a test. You specify a name and location for the data table in the Resources pane of the Test Settings dialog box. For details on the Test Settings dialog box, see "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.

When to save the data table in a different location

- If you want to run the same test with different sets of input values. For example, you can test the localization capabilities of your application by running your test with a different data table file for each language you want to test. You can also vary the user interface strings that you check in each language by using a different environment parameter file each time you run the test. For details, see "Parameterizing Object Values" on page 1526.

- If you need the same input information for different tests. For example, you can test a Web version and a standard Windows version of the same application using different tests, but the same data table file.

When to save a run-time data table

If it is important for you to save the resulting data from the run-time data table, you can insert a DataTable.Export statement to the end of your test to export the run-time data table to a file. You can then import the data to the design-time data table using the data table's **File > Import from File** menu. Alternatively you can add a DataTable.Import statement to the beginning of your test to import the run-time data table that was exported at the end of the previous run session. For more details on these methods, see the *HP UFT Object Model Reference for GUI Testing*.

Saving data tables in an ALM project

When working with ALM, you must save the data table file in the Test Resources module in your ALM project before you specify the data table file in the Resources pane of the Test Settings dialog box. For more details, see "Resources Pane (Test/Business Component Settings Dialog Box) " on page 603.

You can add a new or existing data table file to your ALM project. Note that adding an existing data table file from the file system to an ALM project creates a copy of the file. Thus, once you save the file to the project, changes made to the ALM data table file will not affect the data table file in the file system and vice versa.

Data Table Objects, Methods, and Properties for GUI Testing

Relevant for: GUI tests and scripted GUI components

GUI testing provides several data table methods that enable you to retrieve information on the run-time data table and to set the value of cells in the run-time data table. You enter these statements manually in the Editor. For details on working in the Editor, see "Programming in GUI Testing Documents in the Editor" on page 994.

From a programming perspective, the data table is made up of three types of objects—**DataTable**, **DTSheet** (sheet), and **DTParameter** (column). Each object has several methods and properties that you can use to retrieve or set values.

For more details on the data table methods, see the *HP UFT Object Model Reference for GUI Testing*.

Formulas in Data Tables for GUI Testing

Relevant for: GUI tests and scripted GUI components

You can use Microsoft Excel formulas in your data table. This enables you to create contextually relevant data during the run session. You can also use formulas as part of a checkpoint to check that objects created on-the-fly (dynamically generated) or other variable objects in your Web page or application have the values you expect for a given context.

When you use formulas in a data table to compare values (generally in a checkpoint), the values you compare must be of the same type, for example integers, strings, and so forth. When you extract values from different places in your applications using different functions, the values may not be of the same type. Although these values may look identical on the screen, a comparison of them will fail, since, for example, 8.2 is not equal to "8.2".

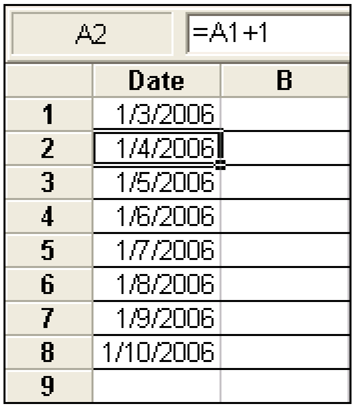
For more details on using worksheet functions, see the Microsoft Excel documentation.

Formulas in Data Tables for Use in Iterations

You can enter formulas rather than fixed values in the cells of a parameter column.

For example, suppose you want to parameterize the value for a WebEdit object that requires a date value no earlier than today's date. You can set the cells in the **Date** column to the date format, and enter the =NOW() Excel formula into the first row to set the value to today's date for the first iteration.

Then you can use another formula in the remaining rows to enter the above date plus one day, as shown below. By using this formula you can run the test on any day and the dates will always be valid.



A2 | =A1+1

	Date	B
1	1/3/2006	
2	1/4/2006	
3	1/5/2006	
4	1/6/2006	
5	1/7/2006	
6	1/8/2006	
7	1/9/2006	
8	1/10/2006	
9		

TEXT or VALUE Functions Used to Convert Values from One Type to Another

- **TEXT(value, format).** Returns the textual equivalent of a numeric value in the specified format, so that, for example the formula =TEXT(8.2, "0.00") is "8.20".

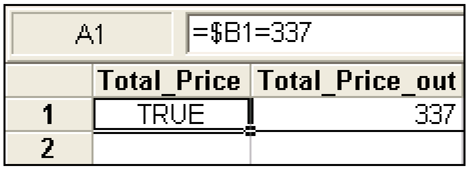
- **VALUE(string).** Returns the numeric value of a string, so that, for example, =VALUE("$8.20") is 8.20.

For more details on using parameters, see "Parameterizing Object Values" on page 1526.

Formulas in Data Tables for Use in Checkpoints

You can use a formula in a checkpoint to confirm that an object created on-the-fly (dynamically generated) or another variable object in your Web page or application contains the value it should for a given context. For example, suppose a shopping cart Web site displays a price total. You can create a text checkpoint on the displayed total value and use a data table formula to check whether the site properly computes the total, based on the individual prices of the products selected for purchase in each iteration.

When you use the data table formula option with a checkpoint, two columns are created in the data table. The first column contains a default checkpoint formula. The second column contains the value to be checked in the form of an output parameter. The result of the formula is Boolean—TRUE or FALSE.



A1 | =$B1=337

	Total_Price	Total_Price_out
1	TRUE	337
2		

A FALSE result in the checkpoint column during a test run causes the test to fail.

After you finish adding the checkpoint, you can modify the default formula in the first column to perform the check you need.

Tasks

For tasks related to BPT, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

How to Define a Data Table in a GUI Test

Relevant for: GUI tests and scripted GUI components

This task describes the different options for defining a new data table for your test.

This task includes:

- "Add an external data table file" below
- "Import data from a database" below
- "Manually enter or import information in the Data table" below
- "Add a data table file to your ALM project" on the next page

Add an external data table file

Configure the location of the data table in the Resources pane of the Test Settings dialog box (**File > Settings > Resources** node). For details, see "Resources Pane (Test/Business Component Settings Dialog Box) " on page 603.

If you select an external file as your data table, make sure that:

- The column names in the external data table match the parameter names in the test.
- The sheets in the external data table match the action names in the test.

Import data from a database

Import data into a data table sheet using Microsoft Query, or by specifying an SQL statement. For details, see the "How to Import Data Into a GUI Test Using Microsoft Query" on page 233.

Manually enter or import information in the Data table

Edit information in the Data pane by typing directly into the table cells. You can also import data saved in Microsoft Excel, tabbed text file (.txt), or ASCII format.

To import an Excel file:

Right-click in the Data pane and select **File > Import from File** from the Data pane commands.

To import a single sheet or a tabbed text file:

Right-click in the Data pane and select **Sheet > Import > From File** from the Data pane commands.

For more details on Data pane commands, see "Data Pane User Interface (GUI Testing)" on page 239.

Add a data table file to your ALM project

1. Make sure you have an accessible Microsoft Excel file with an .xls or .xlsx extension.
2. In ALM, create a new data table resource and then upload the .xls or .xlsx file you created in the previous step to the project's Test Resources module. For more details, see the *HP Application Lifecycle Management User Guide*.
3. In UFT, in the Test Settings dialog box (**File > Settings > Resources** node), select **Other location** and click the browse button to locate the data table file.
4. Create your test. When you save the test, UFT saves the data table file to the ALM project.

How to Manage Data Tables in a GUI Test

Relevant for: GUI tests and scripted GUI components

This task describes how to manage the data table in your test.

This task includes:

- "Define the number of iterations for an action or test" below
- "Change a column name" below
- "Use an autofill list" on the next page

Define the number of iterations for an action or test

- **For actions**: Open the **Run** tab of the **Action Call Properties** dialog box (right-click an action in the canvas and select **Action Call Properties**).
- **For tests**: Open the **Run** pane of the **Settings** dialog box (**File > Settings > Run** pane).

If you want to prevent UFT from running an iteration on a row when the **Run on all rows** option is selected, you must delete the entire row from the data table. To do this, do one of the following:

- Select the row, right-click in the table, and select **Edit > Delete** from the data table's context menu (or use CTRL+K). This restores the bottom grid line from black to gray.
- Use the Clear option from the table's **Edit** menu (or CTRL+X).
- Select a cell and press **Delete** on the keyboard. The data is deleted from the cells, but the row is not deleted and the black line remains. This means that UFT will run an iteration for this row even though there is no data in it.

Change a column name

For details, see "Change Parameter Name Dialog Box (GUI Testing Data Pane)" on page 248.

Use an autofill list

1. Select the type of autofill list from the "AutoFill Lists Dialog Box (GUI Testing Data Pane)", described on page 247.

2. Enter the first item into a cell in the table (item names are case-sensitive).

3. Fill the cells by doing one of the following:

   - Drag the cursor, from the bottom right corner of the cell up or right to fill the next row or column in the sheet.

   - Highlight the item and press ENTER to automatically fill the next row of the sheet.

   - Highlight the item and press TAB to automatically fill the next column of the sheet.

How to Insert Formulas into Data Tables for Use in Checkpoints in a GUI Test

Relevant for: GUI tests and scripted GUI components

This task describes how to insert formulas into the data table for use in a checkpoint.

To use a formula in a checkpoint:

1. Select the object or text for which you want to create a checkpoint and open the Insert Checkpoint dialog box as described in "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

2. In the **Configure value** area, click **Parameter**.

3. Set the parameter options, as described in the "Parameter Options Dialog Box" on page 1556.

4. Specify your other checkpoint setting preferences as described in the "Checkpoint Properties Dialog Box" on page 1432.

5. Highlight the value in the first (formula) column to view the formula and modify the formula to fit your needs.

6. If you want to run several iterations, add the appropriate formula in subsequent rows of the formula column for each iteration in the test or action.

> **Tip:** You can encode passwords to use the resulting strings as method arguments or data table parameter values. For details, see "Password Encoder Tool" on page 965.
>
> You can also encrypt strings in data table cells using the **Encrypt** option in the Data pane menu. For details, see "Data Menu (GUI Testing Data Pane)" on page 243.

How to Import Data Into a GUI Test Using Microsoft Query

Relevant for: GUI tests and scripted GUI components

You can use Microsoft Query to choose a data source and define a query within the data source. For details on supported versions of Microsoft Query, see the *HP Unified Functional Testing Product Availability Matrix*, available from the UFT Help or the root folder of the Unified Functional Testing DVD.

To choose a data source and define a query in Microsoft Query:

1. Open the "Database Query Wizard" (described on page 249) and select **Create query using Microsoft Query**.

2. When Microsoft Query opens, choose a new or an existing data source.

3. Define a query.

4. In the Finish screen of the Query Wizard, select one of the following:

   - **Exit and return to HP Unified Functional Testing > Finish** to exit Microsoft Query.

   - **View data or edit query in Microsoft Query** > **Finish** to view the data.

   After viewing or editing the data, select **File > Exit and return to HP Unified Functional Testing** to close Microsoft Query and return to UFT.

For details on working with Microsoft Query, see the Microsoft Query documentation

How to Add Data Sources to an API Test

For details about data handling for GUI tests and the GUI testing Data Pane, see "Data Pane Overview" on page 222.

Relevant for: API testing only

This task describes how to define data sources before selecting a test step. After creating a test step, you link it to an existing data source. The following steps provide you with different options for creating a data source. To create a data source for your properties, you only need to add data using one of these options.

- "Add an Excel data source" on the next page

- "Add an XML data source" on the next page

- "Add a database data source" on page 235

- "Add a local data table" on page 238

Add an Excel data source

1. Make sure the Data pane is visible. If not, select **View > Data**.

2. In the Data pane, click the **New Data Source** down arrow and select **Excel**. The "New/Change Excel Data Source Dialog Box" (described on page 255) opens.

3. Browse to the file and indicate if the first row is a header row.

4. Specify a name for the data source. If you do not provide a name, UFT sets it to the Excel file name after you navigate to the file.

5. Select whether to link to the Excel in its original location or create a local copy. The advantage of making a local copy is that it becomes portable with the test. However, any updates to the data at its original location will not be not reflected in the migrated test.

6. Select **Allow other tools to override the data** option to enable integration with ALM's Data Awareness. This allows you to overwrite the data from the imported Excel file with values from an ALM Data Resource. For details, see "Data Awareness in ALM" on page 715.

   In addition, if you call a test or action with data assigned to its steps, this option allows you to edit the data in the called test or action. For details, see "Actions Using the Data Table" on page 1791.

7. Click **OK**.

8. In the Data pane, select the data source under the **Excel** node. In the right pane, you can view and edit the values.

Tip: You can replace an existing Excel data source with a new file. For user interface details, see "New/Change Excel Data Source Dialog Box" on page 255.

Add an XML data source

1. Make sure the Data pane is visible. If not, select **View > Data**.

2. In the Data pane, click the **New Data Source** down arrow and select **XML**. The "New XML Data Source Dialog Box" (described on page 257) opens.

3. Provide a **Data Source name**.

4. Select the entity upon which to base the data source:

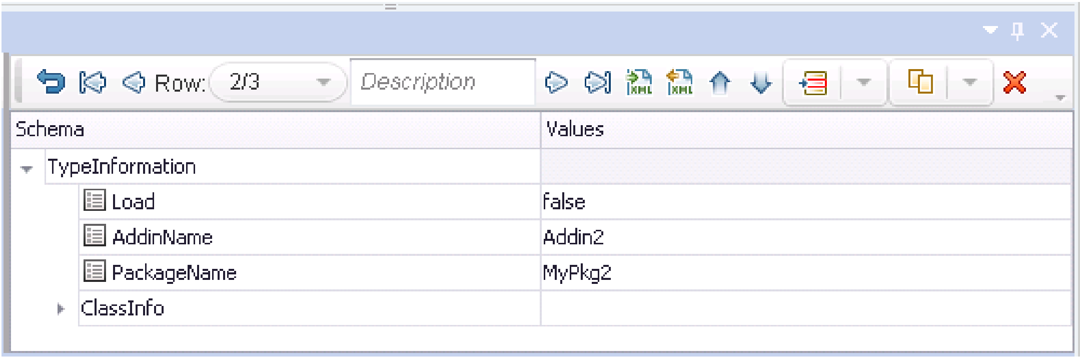
   - **Schema file:** Browse to an .xsd file.

   - **XML file:** Browse to an .xml file.

   - **Use existing**: Browse to an existing XML-based data source from another test.

5. Click **OK**.

6. Select the data source's node in the Data pane.



Schema	Values
TypeInformation	
Load	false
AddinName	Addin2
PackageName	MyPkg2
ClassInfo	

7. Edit the XML data in the grid. Click the **Add rows** button to add row values. For user interface details, see "XML Data Source - Toolbar" on page 253.

8. To load XML values, click the **Load data from an XML file** button.

Add a database data source

1. Make sure the Data pane is visible. If not, select **View > Data**.

2. In the Data pane, click the **New Data Source** down arrow and select **Database**. The "Add New Database Data Source Wizard" (described on page 258) opens. The **Set Database Connection** page opens.

3. For an **OleDB** type connection:

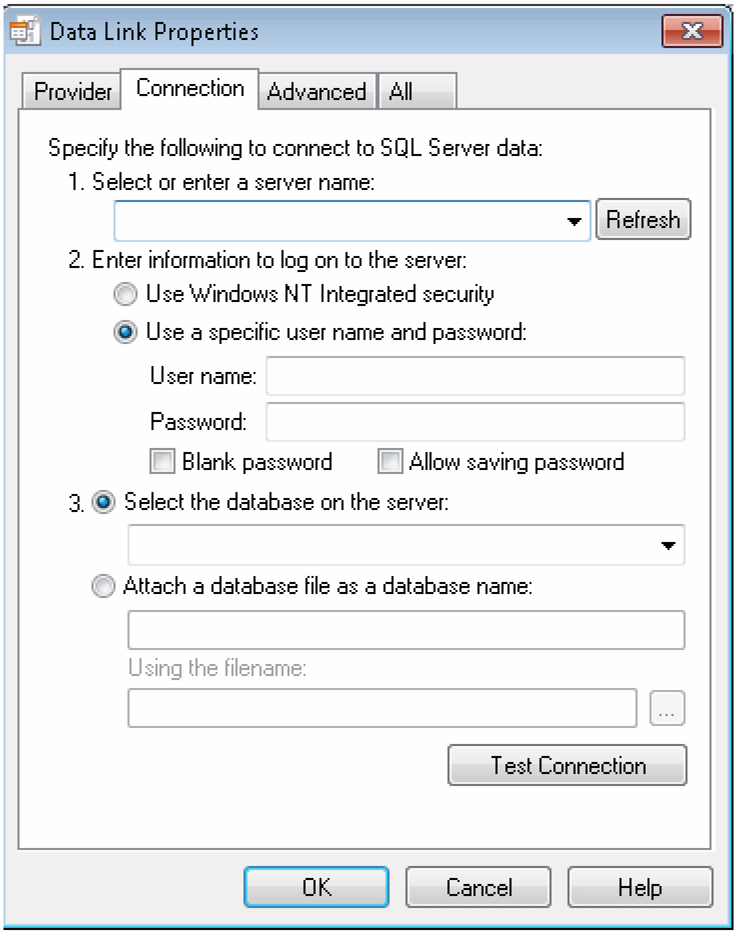
   a. Select **OleDB** as the **Connection type**.

   b. Set the connection string in one of the following ways:

      - Paste an OLE DB connection string into the text area.

      - Select a previously defined connection string from the drop down.

- Click the **Build Connection String** button to use Microsoft's Data Link Properties dialog box. For details, click the **Help** button in the Data Link Properties dialog box.



Specify the following to connect to SQL Server data:

1. Select or enter a server name:

2. Enter information to log on to the server:
   - ○ Use Windows NT Integrated security
   - ● Use a specific user name and password:
     - User name:
     - Password:
     - □ Blank password □ Allow saving password

3. ● Select the database on the server:
   - ○ Attach a database file as a database name:
   - Using the filename:

c. Click **Test Connection** to verify that the connection is valid.

d. Click **Next**. This button is only available for valid connections.

4. For an ODBC type connection:

   a. Select **ODBC** as the **Connection type**.

   b. Click the **Build Connection String** button to select a DSN type. For details, see "Select Data Source Name Dialog Box" on page 262.

   c. Select a DSN type and provide the necessary credentials. To create or edit a DSN entry, click the **Manage ODBC Data Sources** button. For details, click the **Help** button in the ODBC Data Source Administrator dialog box. After you close the administrator dialog box, click the **Reload DNS list** button to refresh the list.

   d. Test the connection. After the confirmation, click **OK**.

5. In the **Set SQL Statement** page:

   a. Provide a unique name for the data source, not used by another data source. If you do not provide a name, the data source is automatically assigned the table name after you build the query.

   b. Provide an SQL statement in one of the following ways:

      - Paste an existing statement into the text area.

      - Select a previously defined statement from the drop down history.

      - Click the **Build Query Statement** button to open the "Query Designer Dialog Box" (described on page 1724). The Query Designer allows you to create a basic SQL query, with a single table or view. For complex queries, such as those using advanced SQL features and calculations, create the query in your environment and paste it into the text area.

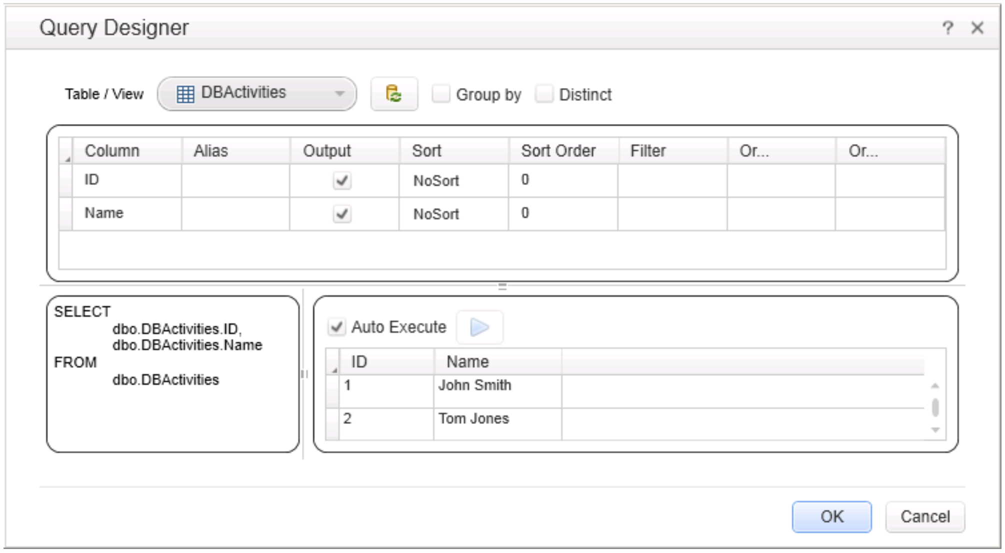


Table / View: DBActivities ☐ Group by ☐ Distinct

Column	Alias	Output	Sort	Sort Order	Filter	Or...	Or...
ID		✓	NoSort	0			
Name		✓	NoSort	0			

```
SELECT
        dbo.DBActivities.ID,
        dbo.DBActivities.Name
FROM
        dbo.DBActivities
```

☑ Auto Execute

ID	Name
1	John Smith
2	Tom Jones

   c. If you are using the **Query Designer**:

      - Select a table or view. The Query Designer lists all of the rows in the pane below and displays the corresponding SQL statement in the preview pane.

      - Enable **Auto execute** to view the results of your query as you make changes in the upper pane.

      - By default, the Query Designer selects all columns in the table. To remove a column from the query, clear the **Output** check box for that column.

      - To set a sorting order, select Ascend or Descend from the **Sort** column drop-down. If you have multiple rows that you are sorting, provide the **Sort Order** beginning with 1. The Query Designer adds an ORDER BY clause to the preview pane.

- To filter the returned results, enter the text to match in the **Filter** column.

- To add a GROUP BY or DISTINCT clause to your statements, select the appropriate check box.

  Click **OK** to accept the query and return to the wizard. Click the **Reset Query** button to discard the changes.

d. In the wizard's **Set SQL Statement** page, click the **Check SQL Statement** button. A Query Preview dialog box opens. Click **Close** to return to the wizard.

e. Click **Finish**. UFT shows the query details in the Properties pane's "Database Data Source Properties Tab (Properties Pane - API Testing)" (described on page 408). By default the Data pane shows the first ten rows of the query data. To change the number of displayed rows, use the "General Pane (Options Dialog Box > API Testing Tab)" (described on page 565).

f. To update the data at a later stage, click **Refresh** in the Data pane. The data is not automatically refreshed—the displayed data is the result of the query when you created the data source or the last time you clicked the **Refresh** button.

Add a local data table

1. Make sure the **Data** pane is visible. If not, select **View > Data**.

2. In the Data pane, click the **New Data Source** down arrow ▾ and select **Local Table**. The "New Local Table Data Source Dialog Box" (described on page 264) opens.

3. Click the **Add** ✚ button to create a column in the data table.

4. In the columns list, specify a column name and description. Select a data type from the drop down list.

5. Repeat steps **3** and **4** for each column you want to create.

6. Use the arrows to reorder the columns as required ⇧ .

7. Click the **Remove** ✖ button to delete an unwanted column from the data table.

8. Click **OK**.

9. In the Data pane, select the table's node in the left pane and move within the table using the keyboard arrows. Type within the cells to manually set values.

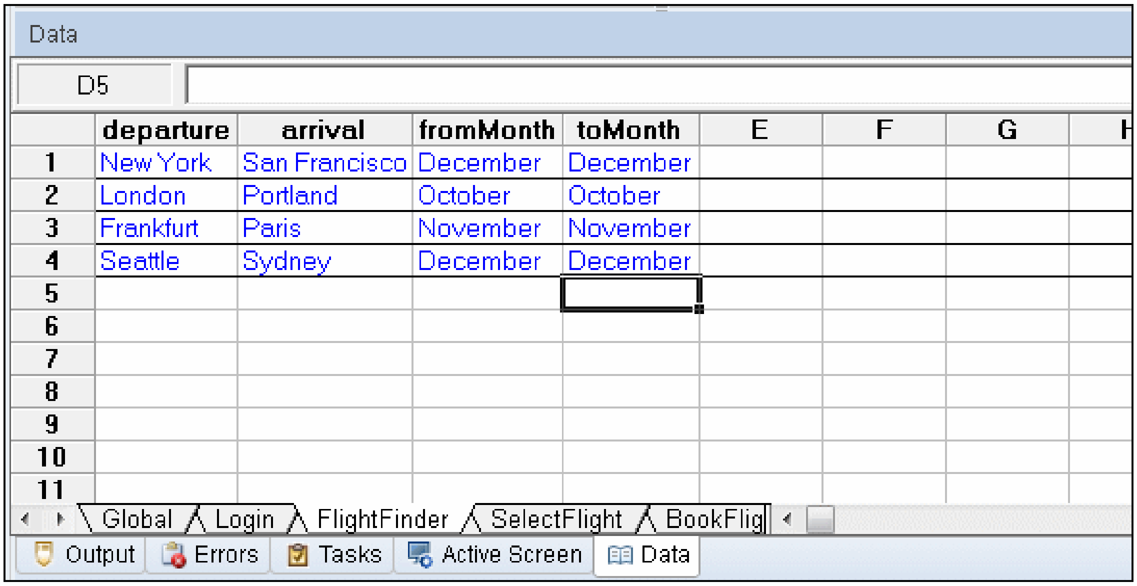
Reference

Data Pane User Interface (GUI Testing)

For details about the API testing Data Pane interface, see "Data Pane User Interface (API Testing) " on page 252. For details about the BPT Data pane interface, see "Data Pane User Interface (BPT in UFT)" on page 265.

Relevant for: GUI tests and scripted GUI components

You use the Data pane in the same way as a Microsoft Excel spreadsheet, including inserting formulas into cells.



	departure	arrival	fromMonth	toMonth	E	F	G	H
1	New York	San Francisco	December	December				
2	London	Portland	October	October				
3	Frankfurt	Paris	November	November				
4	Seattle	Sydney	December	December				
5								
6								
7								
8								
9								
10								
11								

To Access	1. 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. 2. Do one of the following: ▪ Select **View > Data**. ▪ Click the **Data** 📖 toolbar button.

Important information	• You access the Data pane menus and commands by right-clicking anywhere in the Data pane. • Each **row** in the table represents the set of values that UFT submits for the parameterized arguments during a single iteration of the test or action. • Each **column** in the table represents the list of values for a single parameterized argument. The column header is the parameter name. • Combo box and list cells, conditional formatting, and other special cell formats are not supported in the Data pane. • Special characters, machine-dependent characters, and platform-dependent characters are not supported for use in Data Pane column names. • You can also enter data and formulas in cells in the columns that are not intended for use with data table parameters (the columns that do not have a parameter name in the column header).
Relevant tasks	• "How to Define a Data Table in a GUI Test" on page 230 • "How to Manage Data Tables in a GUI Test" on page 231 • "How to Insert Formulas into Data Tables for Use in Checkpoints in a GUI Test" on page 232
See also	• "Data Table Parameters" on page 1533 • For details on supported versions of Microsoft Excel, see the *HP Unified Functional Testing Product Availability Matrix*, available from theUFT Help or the root folder of the Unified Functional Testing DVD.

The Data pane provides the following shortcut menu (right-click) commands:

- "General Commands (GUI Testing Data Pane)" on the next page
- "File Menu (GUI Testing Data Pane)" on the next page
- "Sheet Menu (GUI Testing Data Pane)" on page 242
- "Edit Menu (GUI Testing Data Pane)" on page 242
- "Data Menu (GUI Testing Data Pane)" on page 243
- "Format Menu (GUI Testing Data Pane)" on page 244

This section also includes:

- "Data Pane Specifications (GUI Testing Data Pane)" on page 245
- "Guidelines for Working with the GUI Testing Data Pane" on page 246
- "AutoFill Lists Dialog Box (GUI Testing Data Pane)" on page 247
- "Change Parameter Name Dialog Box (GUI Testing Data Pane)" on page 248

General Commands (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

The **General** shortcut (right-click) menu includes the following commands:

Command	Shortcut Key	Description
Switch between Data pane sheets	CTRL+PAGE UP/PAGE DOWN	Switches through the Data pane sheets when the Data pane is in focus.

File Menu (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

The **File** shortcut (right-click) menu includes the following commands:

Command	Description
Import From File	Imports an existing Microsoft Excel into the Data pane. This command will import all the sheets in the selected Microsoft Excel file. If you want to import only one sheet from an existing Microsoft Excel file or a tabbed text file, use the **Sheet > Import > From File** command described below. **Note:** • The table file you import replaces all data in all sheets of the table. The first row in each Microsoft Excel sheet also replaces the column headers in the corresponding Data pane sheet. It is therefore essential that the first row of your Microsoft Excel sheet exactly matches the parameter names in your test, and that the file contains at least the same number of sheets as the current data table. • If you import a Microsoft Excel table containing combo box or list cells, conditional formatting, or other special cell formats, the formats are not imported and the cells are displayed in the data table with a fixed value.
Export	Exports the table to a specified Microsoft Excel (.xls or .xlsx) file.
Print	Prints the entire table or the selected sheet.

Sheet Menu (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

The **Sheet** shortcut (right-click) menu includes the following commands:

Command	Description
Import > From File	Imports a tabbed text file or a single sheet from an existing Microsoft Excel file into the table. **Note:** The sheet you import replaces all data in the currently selected sheet of the table, and the first row in the Excel sheet replaces the column headers in the corresponding Data pane sheet. It is therefore essential that the first row of your Microsoft Excel sheet exactly matches the parameter names in your test.
Import > From Database	Imports data from the specified database to the current sheet.
Export	Exports the current sheet of the data table to a specified Microsoft Excel (.xlsor .xlsx) file.

Edit Menu (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

The **Edit** shortcut (right-click) menu includes the following commands:

Command	Shortcut Key	Description
Cut	CTRL+X	Cuts the table selection and places it on the Clipboard.
Copy	CTRL+C	Copies the table selection and places it on the Clipboard.
Paste	CTRL+V	Pastes the contents of the Clipboard to the current table selection.
Clear > All	N/A	Clears contents and formatting from the current selection.
Clear > Formats	N/A	Clears the formatting from the current selection.
Clear > Contents	CTRL+DEL	Clears the contents from the current selection.

Command	Shortcut Key	Description
Insert	CTRL+I	Inserts empty cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells. Note that this option is available only when a row or column heading is selected.
Delete	CTRL+K	Deletes the entire current row or column selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells. Note that this option is available only when a row or column heading is selected.
Fill Right	CTRL+R	Copies data in the left-most cell of a selected range to all the cells to the right of that left-most cell within the selected range.
Fill Down	CTRL+D	Copies data in the top cell of a selected range to all cells below that top cell within the selected range.
Find	CTRL+F	Finds a cell containing specified text. You can search by row or column in the table and specify to match case and/or find entire cells only. You can also search for formulas or values.
Replace	CTRL+H	Finds a cell containing specified text and replaces it with different text. You can search by row or column in the table and specify to match case and/or to find entire cells only. You can also search for formulas or values. You can also replace all instances of the found text.
Go To	N/A	Goes to a specified cell. This cell becomes the active cell. You must enter the column and row number of the cell.

Data Menu (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

The **Data** shortcut (right-click) menu includes the following commands:

Command	Shortcut Key	Description
Recalc	F9	Recalculates any formula cells in the table.
Sort	N/A	Sorts a selection of cells by row or column and keys in ascending or descending order.
AutoFill List	N/A	Opens the AutoFill Lists dialog box (described on page 247).

Command	Shortcut Key	Description
Encrypt	N/A	Encodes the text in the selected cells. Note that you cannot decrypt data that has been encrypted. You can also use the Password Encoder to encrypt any text string. This can be useful for entering encrypted strings as method arguments in the Editor. For details, see "Password Encoder Tool" on page 965.

Format Menu (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

The **Format** shortcut (right-click) menu includes the following commands:

Command	Description
General	Sets format to General. The General format displays numbers with as many decimal places as necessary and no commas.
Currency (0)	Sets format to currency with commas and no decimal places. Note: UFT uses the currency symbol defined in your Windows Regional Settings dialog box.
Currency (2)	Sets format to currency with commas and two decimal places. Note: UFT uses the currency symbol defined in your Windows Regional Settings dialog box.
Fixed	Sets format to fixed precision with commas and no decimal places.
Percent	Sets format to percent with no decimal places. Numbers are displayed as percentages with a trailing percent sign (%).
Fraction	Sets format to fraction in numerator denominator form, e.g. 1/2.
Scientific	Sets format to scientific notation with two decimal places.
date (dynamic)	Sets format to Date with the M/D/YY format.
Time: h:mm AM/PM	Sets format to Time with the h:mm AM/PM format.
Custom Number	Sets format to a custom number format that you specify. This option enables you to set special and customized formats for percentages, currencies, dates, times, and so forth.

Data Pane Specifications (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

The main limitations for working with the Data pane are listed below:

Item	Details
Maximum worksheet size	65,536 rows by 256 columns
Maximum number of worksheets	256 (255 sheets in addition to the Global data table).
Column width	0 to 255 characters
Text length	16,383 characters
Formula length	1024 characters
Number precision	15 digits
Largest positive number	9.99999999999999E307
Smallest positive number	1E-307
Largest negative number	-1E-307
Smallest negative number	-9.99999999999999E307
Maximum number of names per workbook	Limited by available memory
Maximum length of name	255
Maximum length of format string	255
Maximum number of tables (workbooks)	Limited by system resources (windows and memory)
Limitations	• The use of colors and formatting in the Data pane is not supported. • Combo box and list cells, conditional formatting, and other special cell formats are not supported in the Data pane.

Guidelines for Working with the GUI Testing Data Pane

Relevant for: GUI tests and scripted GUI components

- When you add data to the Data pane, you must enter the data in rows from top to bottom and left to right—you cannot leave a gap of an entire row or column. For example, if there is data in row 1, you cannot enter data in a cell in row 3 until you have entered data in row 2. Similarly, if there is data in column A, you cannot enter data in column C until you have entered data in column B.

- The value returned from the data table is always converted to a string. If you want the value to be converted to something other than a string, you can use VBScript conversion functions, such as **CInt**, **CLng**, **CDbl,** and so forth. For example:

  ```
  Window("Flight Reservation").WinComboBox("Fly From:").Select CInt(DataTable("ItemNumber", dtGlobalSheet))
  ```

- When you add content to a Data pane cell, the color of the row's bottom grid line changes from gray to black. When you run your test using the **Run on all rows** option (defined in **File > Settings > Run** pane, or the **Run** tab of the "Action Call Properties Dialog Box" (described on page 895), UFT runs one iteration for each row whose bottom grid line is black. For details on modifying the number of iterations, see "How to Manage Data Tables in a GUI Test" on page 231.

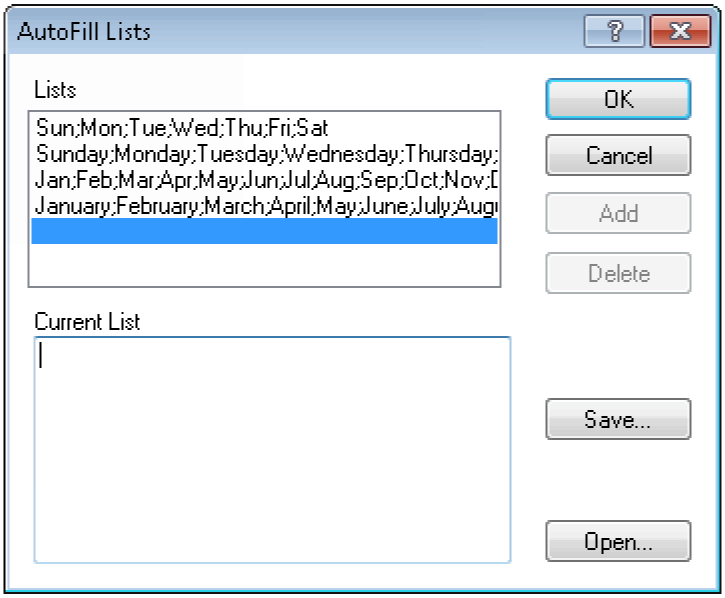
- If you change a data parameter (column header), you must also update the relevant the name of the corresponding parameter wherever it is used in UFT, for example, parameterized argument values, checkpoint or output values, action parameters, and repository parameters.

  If your test is stored in ALM, you must also update the relevant ALM parameter mappings in each configuration defined for your test. For details, see "Change Parameter Name Dialog Box (GUI Testing Data Pane)" on page 248.

AutoFill Lists Dialog Box (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to create, edit, or delete an autofill list. An autofill list contains frequently-used series of text such as months and days of the week.



Sun;Mon;Tue;Wed;Thu;Fri;Sat
Sunday;Monday;Tuesday;Wednesday;Thursday;
Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;D
January;February;March;April;May;June;July;Augu

To Access	In the Data pane (described on page 239), right-click a cell or cell range and select **Data > AutoFill List**.
Important information	AutoFill items are case-sensitive.
Related Tasks	"How to Manage Data Tables in a GUI Test" on page 231
See Also	"Data Pane Overview" on page 222

User-interface elements are described below:

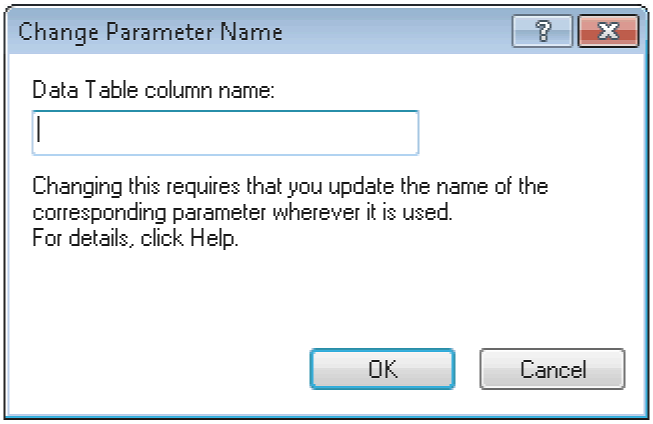
UI Element	Description
Lists	The lists that are available in your project. Four default lists are included.
Current List	The selected list. This pane can be used to create a new list. Separate the items in a new list with a semi-colon.
Add	Adds a new list to the **Lists** box.
Delete	Deletes a list from the **Lists** box.

UI Element	Description
Open	Opens the Open dialog box, in which you can browse to a previously created list.
Save	Opens the Save As dialog box, in which you can save a new list.

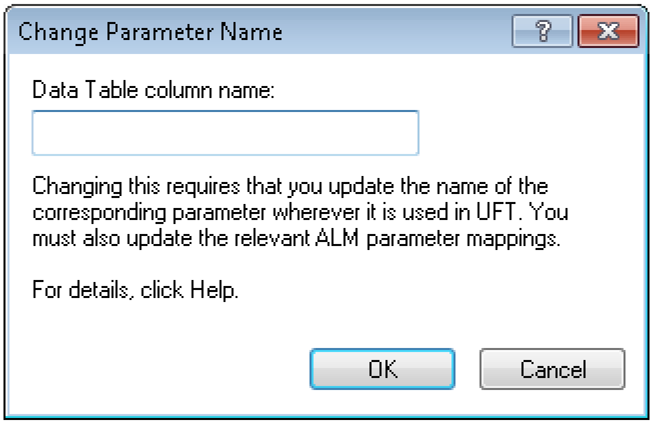
Change Parameter Name Dialog Box (GUI Testing Data Pane)

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to change the Data pane parameter name, which is defined in the column header in the Data pane.



Changing this requires that you update the name of the corresponding parameter wherever it is used.
For details, click Help.



Changing this requires that you update the name of the corresponding parameter wherever it is used in UFT. You must also update the relevant ALM parameter mappings.

For details, click Help.

To access	In the Data pane (described on page 239), double-click a column header cell.

User-interface elements are described below:

UI Element	Description
Data pane column name	The Data pane parameter name, which is defined in the column header in the Data pane. For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269. **Note:** If you modify the column header, you must also: • Change the name of the corresponding data table parameter wherever it is used (for example, parameterized argument values, checkpoint or output values, action parameters, and repository parameters). • **For tests stored in HP ALM:** In the Test Plan module, you must also update the parameter mappings in each configuration defined for your test.

Database Query Wizard

Relevant for: GUI tests and scripted GUI components

This wizard enables you to import data from a database to a Data pane sheet.

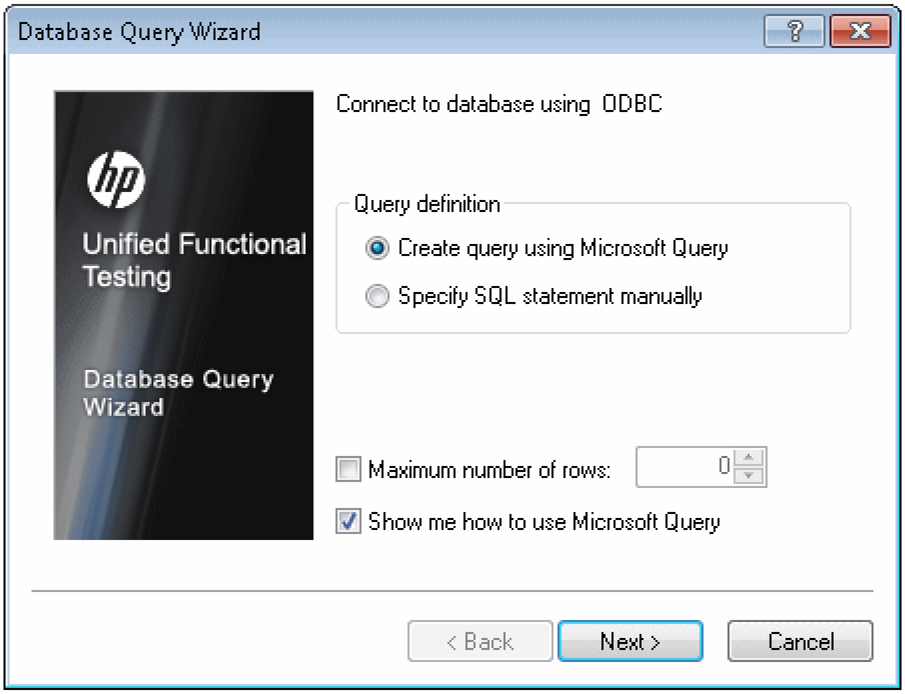
To access	In the "Data Pane User Interface (GUI Testing)" (described on page 239), right-click the sheet to which you want to import the data and select **Sheet > Import > From Database**.
Important information	• You can install Microsoft Query from the custom installation option of Microsoft Office. • UFT takes several seconds to capture the database query and restore the UFT window. The resulting data from the database query is displayed in the Data pane. • Contrary to importing an Excel file (**File > Import From File**), existing data in the Data pane is not replaced when you import data from a database. If the database you import contains a column with the same name as an existing column, the database column is added as a new column with the column name followed by a sequential number. For example, if your data table already contains a column called departures, a database column by the same name would be inserted into the data table as departures1.
Relevant tasks	• "How to Manage Data Tables in a GUI Test" on page 231 • "How to Import Data Into a GUI Test Using Microsoft Query" on page 233

Wizard map	The **Database Query Wizard** contains: **Connect to database using ODBC** (250) **>** **Specify SQL statement** (page 251)

Connect to Database Using ODBC Page (Database Query Wizard)

Relevant for: GUI tests and scripted GUI components

This wizard page enables you to select a data source to import from.



Connect to database using ODBC

Query definition

● Create query using Microsoft Query

○ Specify SQL statement manually

□ Maximum number of rows: 0

☑ Show me how to use Microsoft Query

Wizard map	The "Database Query Wizard" contains: **Connect to database using ODBC** > Specify SQL statement

User interface elements are described below:

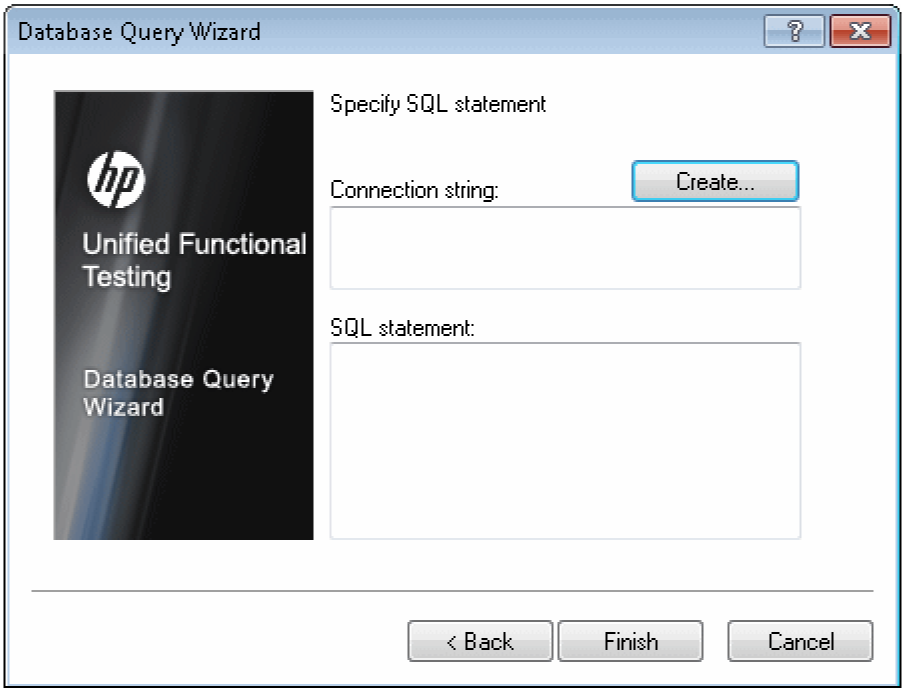
UI Elements	Description
Create query using Microsoft Query	Opens Microsoft Query, enabling you to create a new query. After you finish defining your query, you exit back to UFT. This option is only available if you have Microsoft Query installed on your computer.
Specify SQL statement manually	Opens the **Specify SQL statement** page in the wizard, which enables you to specify the connection string and an SQL statement.

UI Elements	Description
Maximum number of rows	The maximum number of database rows to import. You can specify a maximum of 32,000 rows.
Show me how to use Microsoft Query	Displays an instruction screen before opening Microsoft Query when you click **Next**. (Enabled only when **Create query using Microsoft Query** is selected).

Specify SQL Statement Page (Database Query Wizard)

Relevant for: GUI tests and scripted GUI components

This wizard page enables you to define the SQL statement to import into the Data pane.



Wizard map	The "Database Query Wizard" contains: Connect to database using ODBC > **Specify SQL statement**

User interface elements are described below:

UI Elements	Description
Connection string	The connection string to import.

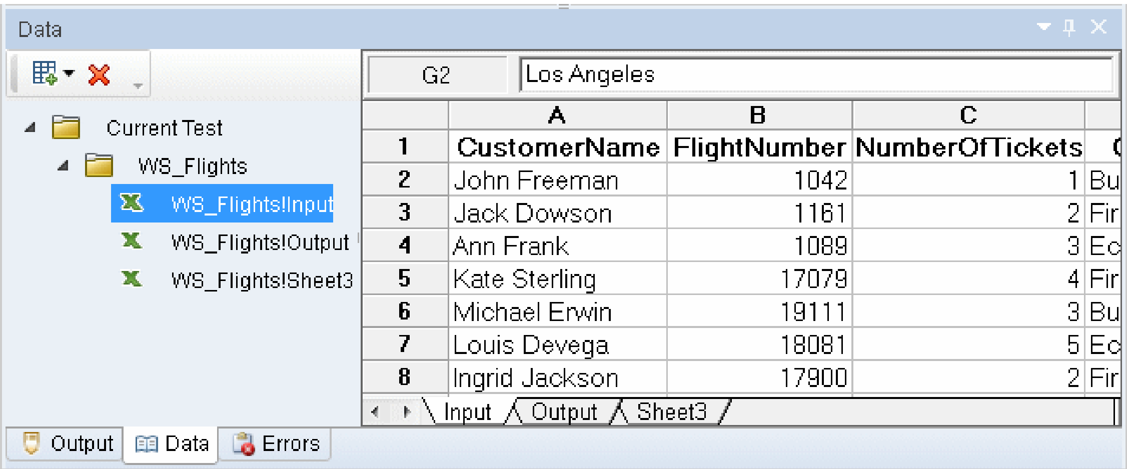
UI Elements	Description
Create	Opens the ODBC Select Data Source dialog box. You can select a .dsn file in the ODBC Select Data Source dialog box or create a new .dsn file to have it insert the connection string in the box for you.
SQL statement	The details of the SQL statement.

Data Pane User Interface (API Testing)

For details about the GUI testing Data Pane interface, see "Data Pane User Interface (GUI Testing)" on page 239. For details about the BPT Data Pane interface, see "Data Pane User Interface (BPT in UFT)" on page 265.

Relevant for: API testing only

The Data pane enables you to add, view, and edit data sources for use as property values when running tests.



G2 Los Angeles

	A	B	C	
1	CustomerName	FlightNumber	NumberOfTickets	(
2	John Freeman	1042	1	Bu
3	Jack Dowson	1161	2	Fir
4	Ann Frank	1089	3	Ec
5	Kate Sterling	17079	4	Fir
6	Michael Erwin	19111	3	Bu
7	Louis Devega	18081	5	Ec
8	Ingrid Jackson	17900	2	Fir

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component 2. Select **View > Data** or select the Data pane tab.
Important information	In order to see the contents of a data source, click its node in the left pane.
Relevant tasks	"How to Add Data Sources to an API Test" on page 233

See also	• "How to Assign Data to API Test/Component Steps" on page 1819 • "How to Set the Data Source Navigation Properties" on page 1825

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

UI Element	Description
<data source tree>	A hierarchy of the data sources and their subnodes, displayed in the left pane. There are separate nodes for the test and action data sources: Current Test, <action1>, <action2>, and so forth. For Excel files, each sheet represents another set of properties.
<data source content>	The data associated with the selected node, displayed in the right pane. You can edit values directly from this pane. If the current test calls another test or action with its own data source, this area shows the data, provided that you enabled the data to be viewed as described below. **Note:** To enable this data to be viewed and edited, click the node in the original location of the data, and enable the **Allow other tools to override the data** option. For details, see "Data Source Properties Tab (Properties Pane - API Testing)" on page 406.
▦▾	**New Data Source.** Creates a new a data source. Expand the drop down to select a data source type: **Excel**, **XML**, **Database**, or **Local Table**.
✖	**Remove.** Removes the selected data source from the Data pane.

XML Data Source - Toolbar

The following controls are available when selecting an XML data source in the Data pane.

UI Elements	Description
↩	**Go back.** Returns to the previous view.
⏮	**Go to start.** Returns to the first row.
◁	**Go to the previous row.** Navigates to the previous row (one row up).
Row	The current row number out of the total rows, such as 1/3 and so forth. Use the drop down to move to the desired row.
Description	A title for the row currently displayed.
▷	**Go to the next row.** Navigates to the next row (one row down).

UI Elements	Description
▷|	**Go to end.** Navigates to the last row.
XML	**Import data from an XML file.** Loads data from an XML file.
XML	**Export data from an XML file.** Exports the current data to an XML file.
↑	**Move up.** Moves the current row forwards.
↓	**Move down.** Moves the current row backwards.
☰▾	Adds a new empty row at one of the following locations. • **Add after current** • **Add at start** • **Add at end**
⧉▾	Duplicates the current row at one of the following locations. • **Duplicate after current** • **Duplicate at start** • **Duplicate at end**
✖	Deletes the current row.

Database Data Source Nodes

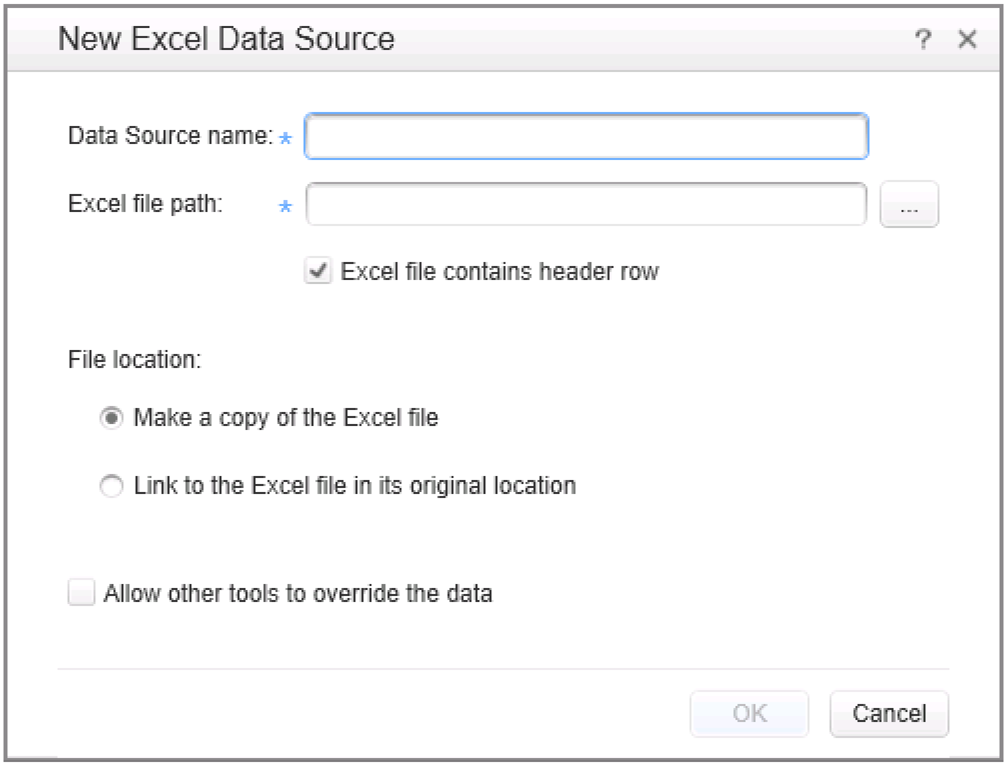
The following controls are available when selecting a database data source in the Data pane.

UI Elements	Description
Refresh	**Refresh.** Updates the data from the database.
Count	**Count.** Opens a message box with a row count. By default the Data pane shows the first ten rows of the query data. To change the number of displayed rows, use the "General Pane (Options Dialog Box > API Testing Tab)"(described on page 565).

New/Change Excel Data Source Dialog Box

Relevant for: API testing only

This dialog box enables you to add a new Excel data source to the Data pane or change the underlying file for existing Excel data sources.



New Excel Data Source

Data Source name: *

Excel file path: *

☑ Excel file contains header row

File location:

● Make a copy of the Excel file

○ Link to the Excel file in its original location

□ Allow other tools to override the data

To access	Before you open the dialog boxes, make sure that an API test or component is in focus in the document pane. **To open the New Excel Data Source dialog box:** 1. Select **View > Data** 2. Click on the **New Data Source** down arrow and select **Excel**. **To open the Change Excel Data Source dialog box:** 1. Select the main node of an existing Excel data source. 2. In the Properties pane, click **Change Excel File**.
Relevant tasks	"Add an Excel data source" on page 234
See also	"Data Link/Relation Message Box" on page 1837

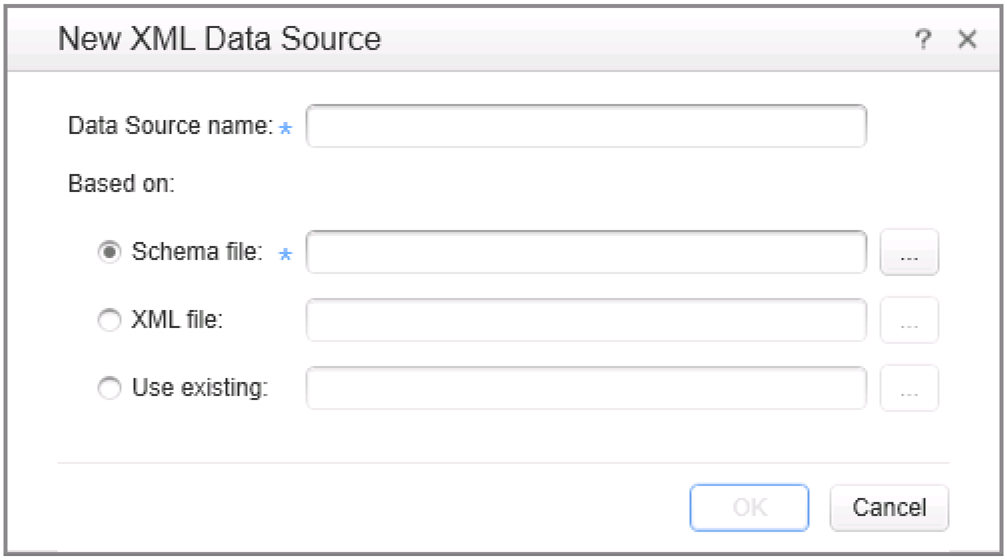
User interface elements are described below:

UI Element	Description
Data source name	A name for the data source as it will appear in the Data pane. If no value is specified, it uses the file name. **Note:** This field is read-only in the Change Excel Data Source dialog box.
Excel file path	The complete path of an Excel file on the file system. Use the **Browse** button to locate the file. **Note:** You must use an absolute path for the Excel file path.
Excel file contains header row	Indicates whether the data in the Excel table contains a header row. Header rows are ignored when running the test.
File location	The location for storing the imported Excel data: • **Link to the Excel file in its original location.** If the data changes at its source, it affects your test. In addition, if you modify the data in your test, it affects the data at its source. • **Make a copy of the Excel file.** Copies the Excel file locally. If the data changes at its source, it does not affect your test. In addition, the data is portable together with the test.
Allow other tools to override the data	Enables you to overwrite data from the imported Excel file with values from other tools, such as an ALM Data Resource (for versions ALM 11.00 and higher), and GUI or API tests. For details about ALM data resources, see "Data Awareness in ALM" on page 715. **Tip:** If you did not enable this option when you imported the Excel file, you can enable it separately for each data source in the Properties pane. For details, see "Data Source Properties Tab (Properties Pane - API Testing)" on page 406.

New XML Data Source Dialog Box

Relevant for: API testing only

This dialog enables you to add new XSD or XML data sources to the Data pane.



To access	1. Do one of the following: ■ Ensure that an API test or component is in focus in the document pane. ■ In the solution explorer, select an API test or component. 2. Create or open a test or component. 3. Select **View > Data**. 4. Click on the **New Data Source** down arrow and select **XML**.
Relevant tasks	"Add an XML data source" on page 234
See also	"XML Data Source - Toolbar" on page 253

User interface elements are described below:

UI Element	Description
Data source name	The display name of the data source. If no name is specified, it uses the file name.

UI Element	Description
Based on	The file on which to base the data table: - **Schema file.** A file with an XSD extension. - **XML file.** A file with an XML extension. - **Use existing.** An existing data resource created within another test. Since this data source is referenced, changes made to it are reflected in all tests using the data source. Use the **Browse** button to navigate to the desired location.

Add New Database Data Source Wizard

Relevant for: API testing only

This wizard enables you to create a data source by retrieving data rows from a database table. You specify a connection string and an SQL statement to retrieve the data.

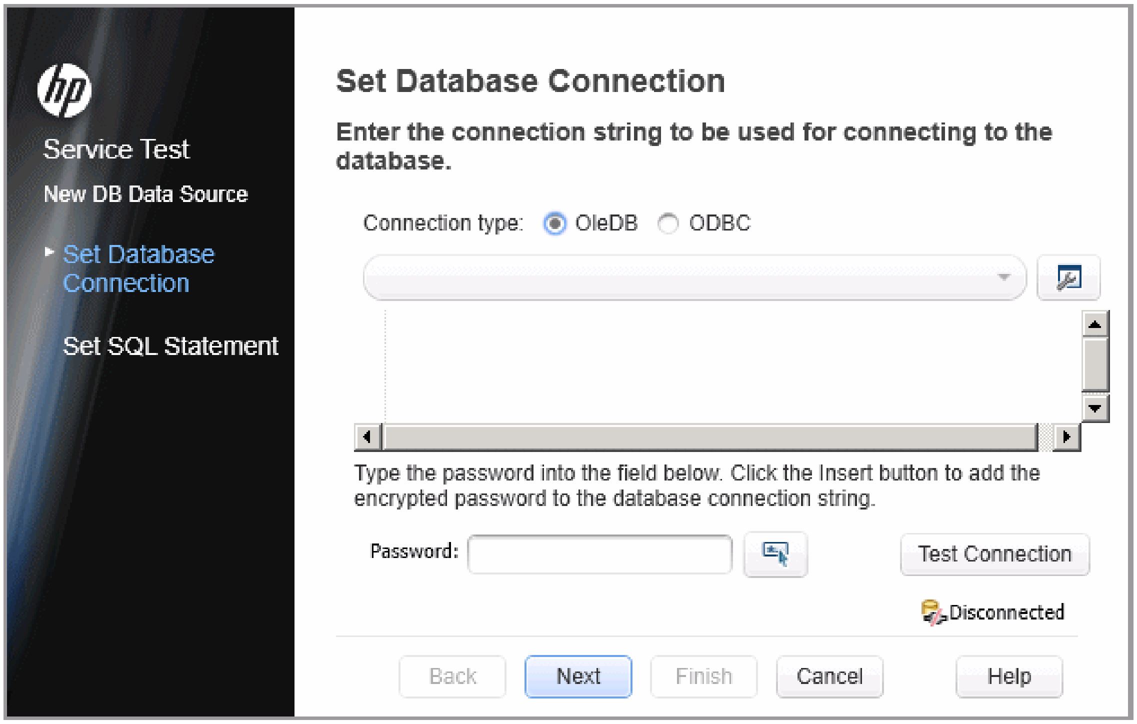
To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. Create or open a test or component. 3. Select **View > Data**. 4. Click on the **New Data Source** down arrow ▦▾ and select **Database**.
Relevant tasks	"Add a database data source" on page 235
Wizard map	This wizard contains: "Set Database Connection Page" > "Set SQL Statement Page"
See also	"Query Builder Dialog Box" on page 1722

Parent Topic: "Data Pane User Interface (API Testing)" on page 252

Set Database Connection Page

Relevant for: API testing only

This wizard page enables you to define a connection string for connecting to the database.



Set Database Connection

Enter the connection string to be used for connecting to the database.

Connection type: ◉ OleDB ○ ODBC

Type the password into the field below. Click the Insert button to add the encrypted password to the database connection string.

Password:

Wizard map	This wizard contains: **Set Database Connection Page** > "Set SQL Statement Page"

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

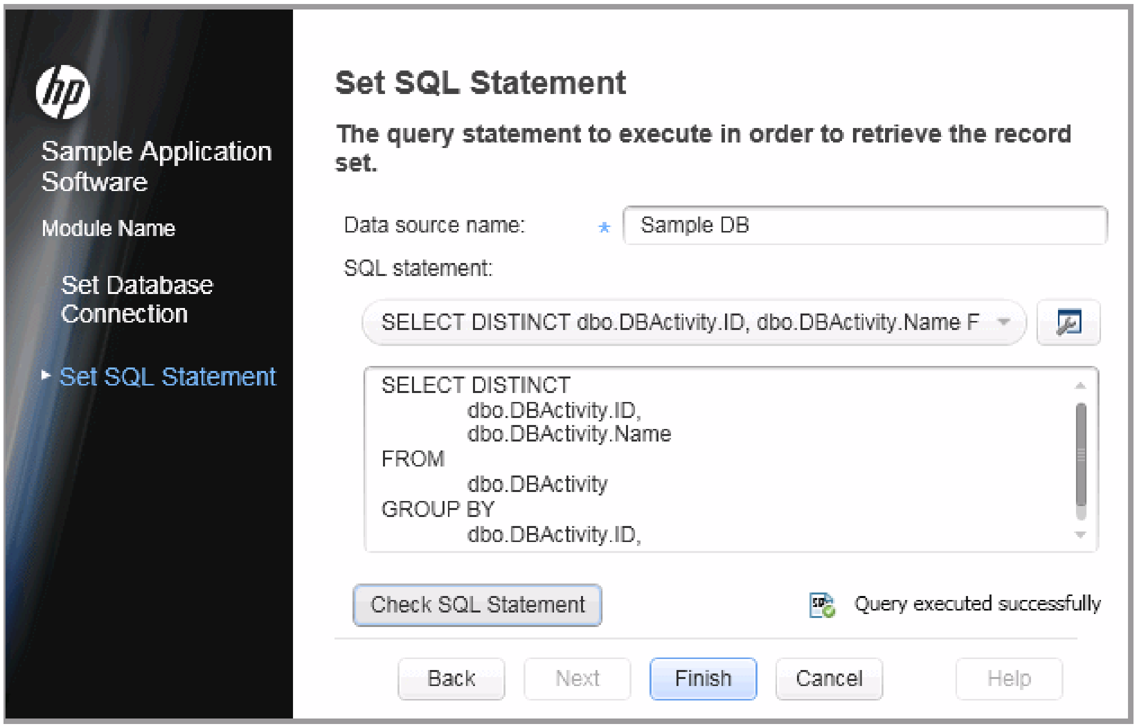
UI Element	Description
Connection type	The type of database connection: **OleDB** or **ODBC**.
<connection list>	A drop down list of previously defined connection strings.
🔧	**Build Connection String.** • For **OleDB** connections: Opens the Microsoft Data Link Properties dialog box. • For **ODBC** connections: Opens the Select Data Source Name dialog box.

UI Element	Description
<connection string area>	An editable area for pasting in existing strings, or for editing the connection string selected in the drop down list. **Tip:** To add a string to the drop down list, click **Check Connection**.
Password	The password with which to access the database.
	Insert. Inserts the encoded password into the displayed connection string.
Test Connection	Sends the current connection string to the database server to check its validity.
<connection status>	Displays the status of the database connection. If it is a valid connection, the wizard adds the connection string to the list of Connection strings and changes the status to Connected Connected .

Set SQL Statement Page

Relevant for: API testing only

This wizard page enables you to prepare an SQL statement for retrieving your data.



Set SQL Statement

The query statement to execute in order to retrieve the record set.

Data source name: * Sample DB

SQL statement:

SELECT DISTINCT dbo.DBActivity.ID, dbo.DBActivity.Name F

```
SELECT DISTINCT
        dbo.DBActivity.ID,
        dbo.DBActivity.Name
FROM
        dbo.DBActivity
GROUP BY
        dbo.DBActivity.ID,
```

Check SQL Statement   Query executed successfully

Wizard map	This wizard contains: "Set Database Connection Page" > **Set SQL Statement Page**

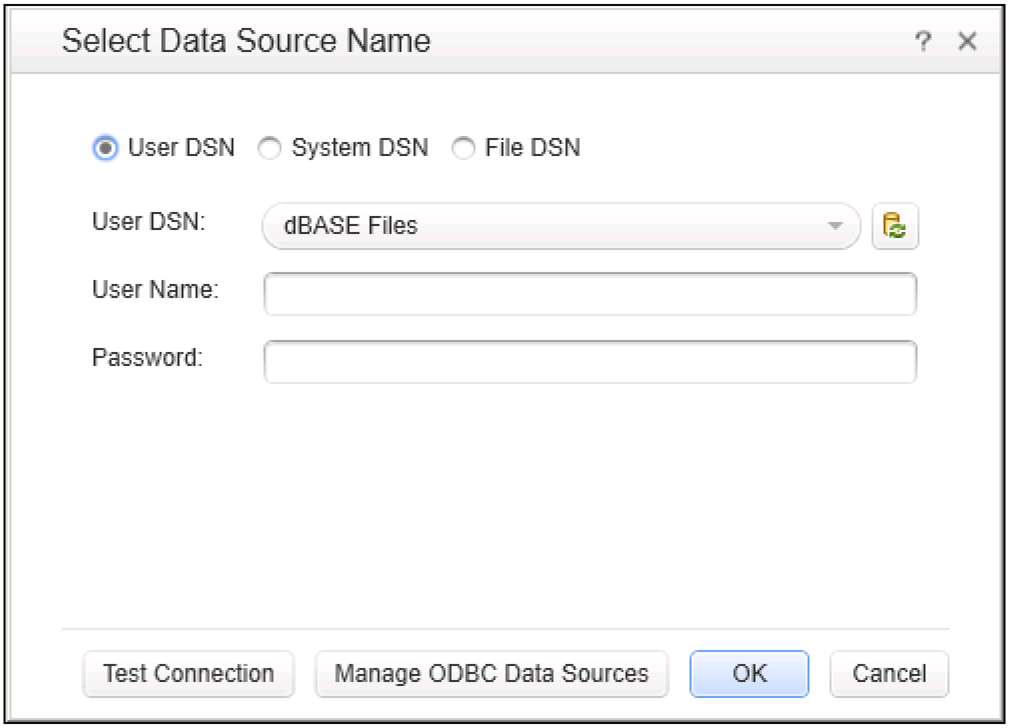
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
Data source name	The name of the data source as it is be referenced by your test. If no value is specified, it uses the name of the table or view.
SQL statement	A drop down list of the available SQL statements.
	Opens the "Query Builder Dialog Box" (described on page 1722).
<SQL statement area>	An editable area for pasting existing SQL strings or for editing the SQL string selected in the drop down list. **Tip:** To add an SQL statement pasted into the text area to the drop down list, click **Check SQL Statement**.
Check SQL Statement	Executes the SQL statement shown in the text area and opens the results in a **Query Preview** window.
<query status>	If the statement is valid, the wizard adds it to the drop down list of SQL statements and changes the status to Query executed successfully.

Select Data Source Name Dialog Box

Relevant for: API testing only

This dialog helps you build a connection string for an ODBC data source.



Select Data Source Name

● User DSN ○ System DSN ○ File DSN

User DSN: dBASE Files

User Name:

Password:

To access	Do one of the following: **From the Database wizard:** 1. Choose **Database** in the Data pane to open the wizard. For details, see "Add New Database Data Source Wizard" on page 258. 2. Select **ODBC** as a connection type. 3. Click on the **Build Connection String** button adjacent to the Connection list drop down list. **From the Properties pane:** 1. Add a Database **Open Connection** activity to the canvas. 2. Click the **Input/Checkpoints** tab in the **Properties** pane and click the browse button in the **Connection string** property row. 3. In the "Connection Builder Dialog Box" (described on page 1720), select **ODBC** as a connection type. 4. Click on the **Build Connection String** button adjacent to the Connection list drop down.
Relevant tasks	"Add a database data source" on page 235
See also	"Add New Database Data Source Wizard" on page 258

User interface elements depend on the type of DSN selected:

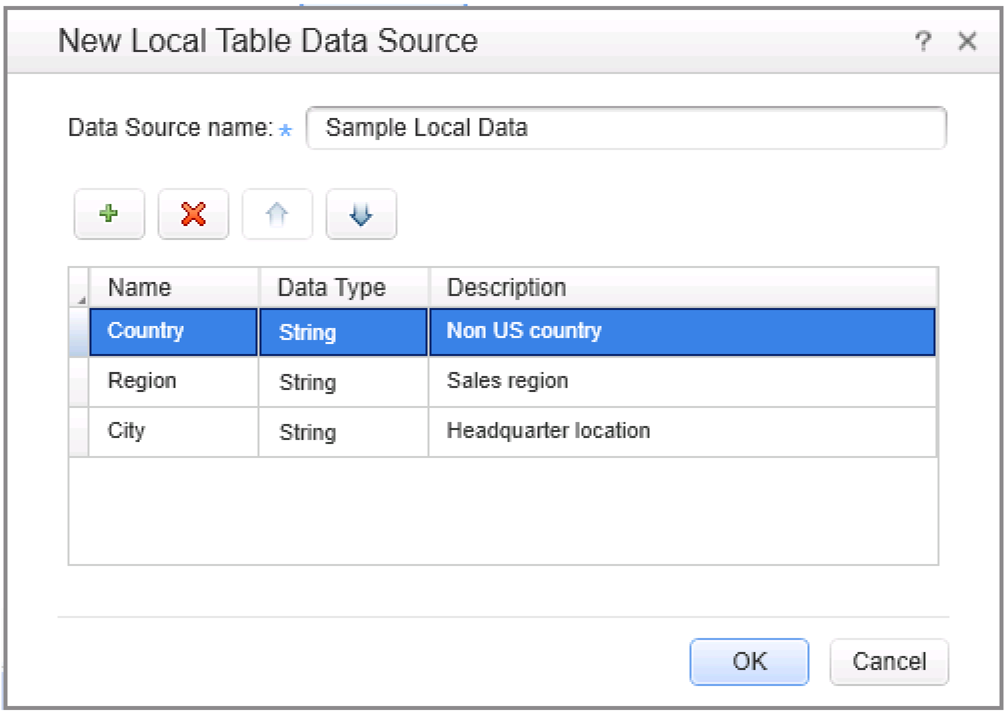
UI Element	Description
🔄	**Reload DSN List**. Reloads the items that will appear in the drop down list.
User DSN	Allows you to select a User DSN and provide credentials: - User Name - Password
System DSN	Allows you to select a System DSN and provide credentials: - User Name - Password

UI Element	Description
File DSN	Allows you to provide the following information for the File DSN or select a file with the required settings. • User Name • Password • File Content
Test Connection	Sends the current DSN information to the database server to check its validity.
Manage ODBC Data Sources	Opens the Microsoft ODBC Data Source Administrator dialog box. This dialog box lets you add, edit or delete data sources at the Windows level. **Tip:** After you create or modify a data source on the Windows level, click the **Reload DSN List** button to refresh the DSN list.

New Local Table Data Source Dialog Box

Relevant for: API testing only

This dialog box enables to create local data tables in the Data pane for use with your test.



Data Source name: * Sample Local Data

Name	Data Type	Description
Country	String	Non US country
Region	String	Sales region
City	String	Headquarter location

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. Create or open a test or component 3. Select **View > Data**. 4. Click on the **New Data Source** down arrow ▦▾ and select **Local Table**.
Relevant tasks	"Add a local data table" on page 238

User interface elements are described below:

UI Element	Description
✚	Adds a new column to the local data table. New entries are added below the last column.
✖	Deletes the selected data table column.
⇧ ⇩	Raises or lowers the selection in the column list.
Columns	A list of the columns in the local data table. Click inside a table row to set or edit the following values: • **Name.** The name of the column. • **Data Type.** A data type: String, Integer, Double, or Date. • **Description.** A description of the data in the column.

Data Pane User Interface (BPT in UFT)

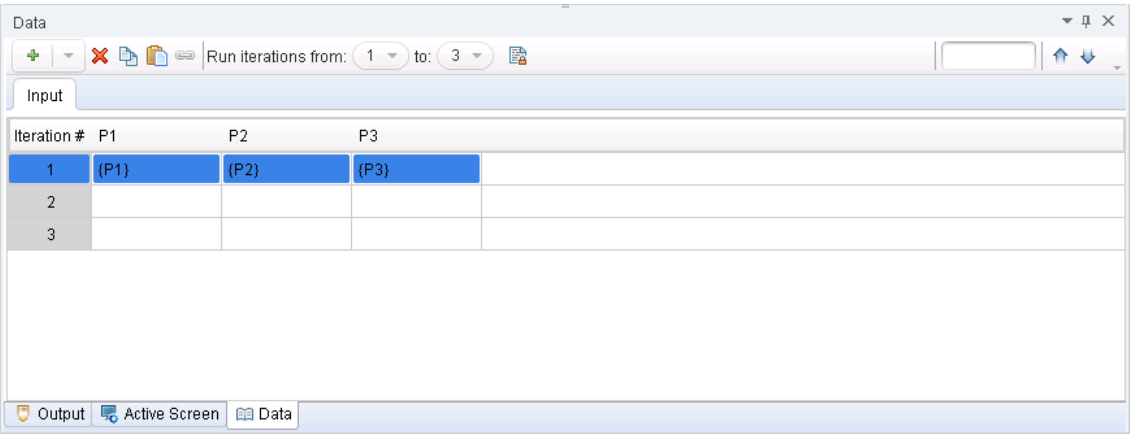
For details about the GUI testing Data Pane interface, see "Data Pane User Interface (GUI Testing)" on page 239. For details about the API testing Data Pane interface, see "Data Pane User Interface (API Testing)" on page 252.

Relevant for: business process tests and flows

The Data pane for business process tests enables you to set the number of iterations to run a selected component or group as well as set the values for the parameters included with the components.

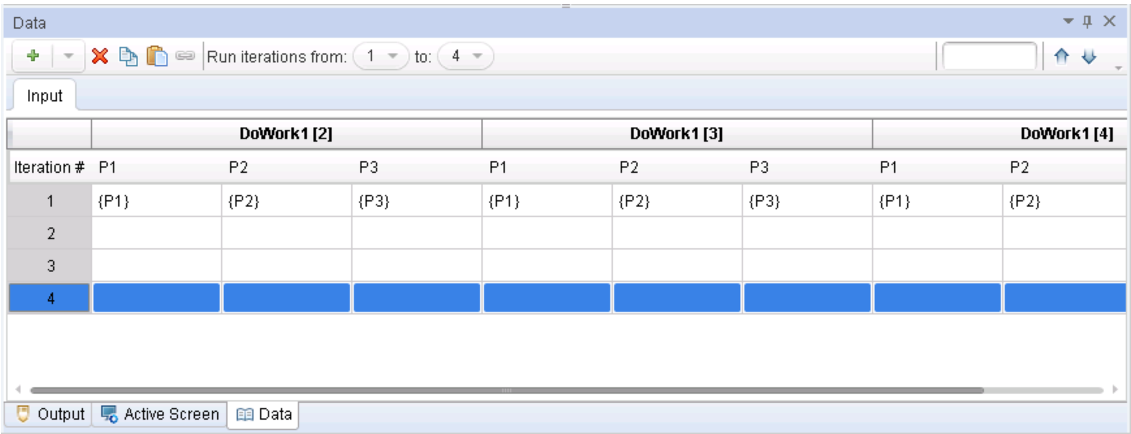
The following image shows an example of the Data pane when a component is selected in the document pane, while editing a test.

If the user was editing a flow, the Data pane would display an additional Output tab for output parameters.



Iteration #	P1	P2	P3
1	{P1}	{P2}	{P3}
2			
3			

The following image shows an example of the Data pane when a group is selected in the document pane. Columns are displayed for each component in the group, with cells for each parameter value in the various iterations.



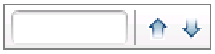
	DoWork1 [2]			DoWork1 [3]			DoWork1 [4]	
Iteration #	P1	P2	P3	P1	P2	P3	P1	P2
1	{P1}	{P2}	{P3}	{P1}	{P2}	{P3}	{P1}	{P2}
2								
3								
4								

To access	1. Ensure that a business process test or flow is in focus in the document pane, and a component, flow, or group is selected. 2. Do one of the following: ▪ Select **View > Data**. ▪ Click the **Data** toolbar button.

Important Information	The first column in the Data pane grid always displays the iteration number. Other columns display data for specific parameters, or if you have a group selected in the document pane, for specific components or flows in the group. After adding iterations, enter text or formulas in the cells to add parameter data for the specific iteration. For more details, see "Considerations for BPT Test Iterations and Parameters" on page 224.
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
+ ▾	**Add New Iteration.** Adds a new iteration without setting any parameter values. Click the drop-down button to perform one of the following tasks: • **Add New Iteration.** Adds a new iteration without setting any parameter values. • **Copy Iteration.** Adds a new iteration by copying the values of a selected iteration's parameters. • **Create Iteration with Default Values.** Adds a new iteration and sets the parameter values to the default values. **Note:** This button is disabled if the component or flow for which you are adding an iteration is in a group or a flow, or, if you are selecting a specific cell in the table. To enable this button, select an iteration row in the grid and make sure that the component or flow selected in the document pane is not in a group.
✖	**Delete.** Deletes the selected iteration. **Note:** This option is disabled when the component or flow for which you are adding an iteration is in a group or a flow.
⎘	**Copy (CTRL+C).** Copies the data in the selected iteration or cell to the clipboard.

UI Element	Description
📋	**Paste (CTRL+V).** Pastes the copied iteration or cell. This action replaces the selected data with the copied data. You can also use this button to paste data from an external source, such as an Excel spreadsheet.
🔗	**Link Parameters.** Opens the Select Link Source dialog box (described on page 2128), which enables you to link parameter values to other parameters in your test or flow. **Note:** This button is disabled if you have an entire iteration row selected in the grid.
Run iterations from:	Enables you to select specific iterations to run. Select the first iteration to run from the first drop-down menu, and then select the last iteration to run from the second drop-down menu.
🔐	**Encrypt.** Encrypts the currently selected parameter.
	**Search toolbar.** Enables you to search the data pane for a specific value. Use the up and down arrows to navigate to the next or previous search result.
Input/Output tabs	Enables you to add and modify iteration data for input or output parameters. The Output tab is available only when editing flows.
<iteration # column>	Indicates the iteration number for the row, in ascending order.
<value columns>	Displays the value for each parameter in the displayed iteration. Test or flow parameters (parameters whose values are to be taken from the test or flow level), are shown in braces **{}**. Enter values directly in the cell or link them using the Select Link Source dialog box. To promote a parameter value to a test or flow parameter, add braces around the value you enter. If no value is specified in the Properties pane Parameters tab, the default value for that parameter is used.

Troubleshooting and Limitations - Data Pane

Relevant for: GUI tests, **scripted GUI components**, and **API testing**.

This section describes troubleshooting and limitations for working with the Data pane.

Incompatible Microsoft Excel Files (GUI tests and components only)

Microsoft Excel files may not be compatible with UFT. This may occur because the .xlsor .xlsx file you are attempting to import contains features or functionality that is unavailable for GUI tests.

Workaround 1: Create and import .xls or .xlsx files that contain only functionality that is available from the GUI testing Data pane itself. For a list of the available functionality in the Data pane, see the relevant shortcut menu in "Data Pane User Interface (GUI Testing)" on page 239.

Workaround 2: If you need to alter an existing .xls or .xlsx file, you can use one of the following options:

- Determine what are the most recent changes to the .xlsor .xlsx file, and undo or remove them.
- Remove any existing macros.
- In each worksheet in the .xls or .xlsx file, select all rows and columns, and select **Edit > Clear > Formats** to remove all formatting from the worksheet.
- Create a new .xls or .xlsx file, and copy the data from the non-compatible file. You can do this by using the standard **copy** and **paste** operations, or you can do this by saving each worksheet as a tab delimited .txt file (**File > Save As**), opening each .txt file in a text editor, and copying the data from that file to a new .xlsor .xlsx file.

> **Caution:**
>
> - Make sure to back up your .xls or .xlsx file before changing or removing any data or functionality.
> - Even though some of the operations in this workaround can be automated, due to the way that Microsoft Excel exports data to tab delimited text files, the actual exported data may not exactly match the original data. Therefore, make sure to verify all exported data.

Importing/Exporting a Data Table

If you import a Microsoft Excel table containing a combo box, list cells, conditional formatting, or other special cell formats, the formats are not imported and the cells are displayed in the data table with fixed values. Make sure to verify all imported data from cells containing any of these special formats because the original formatting may affect the values, and the values may not be imported correctly.

Data Table Cell Content

Entering a very large number in the data table may cause unexpected behavior.

Troubleshooting and limitations for using the Data pane with API tests and components.

- Excel data with column headers beginning with digits or containing symbols (such as **!**, **@**, or **#**) are not supported.
- XML data sources: .xml files that use XSL are not supported.

Chapter 9: Debug Panes

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This chapter includes:

Concepts

Debug Panes Overview

Relevant for: GUI actions, scripted GUI components, function libraries and user code files

After creating a test, component, function library, or user code file, you can check to see if they run properly. Using the debug panes, you can then debug your tests, components, function libraries, or user code files using UFT's debugging capabilities.

The debug panes are the primary interface for viewing information about your application during run sessions, as well as addressing any errors in the run session.

For a conceptual overview of UFT's debugging capabilities, see "Debugging Overview" on page 675.

For a task related to debugging, see "How to Debug Your Test, Component, Function Library, or User Code File" on page 683.

For exercises to practice using UFT's debugging capabilities, see :

- "How to Debug a GUI Action or a Function - Exercise" on page 689
- "How to Debug an API User Code File - Exercise" on page 693
- "How to Step Into, Out of, or Over a Specific Step in a GUI Test - Exercise" on page 697

Reference

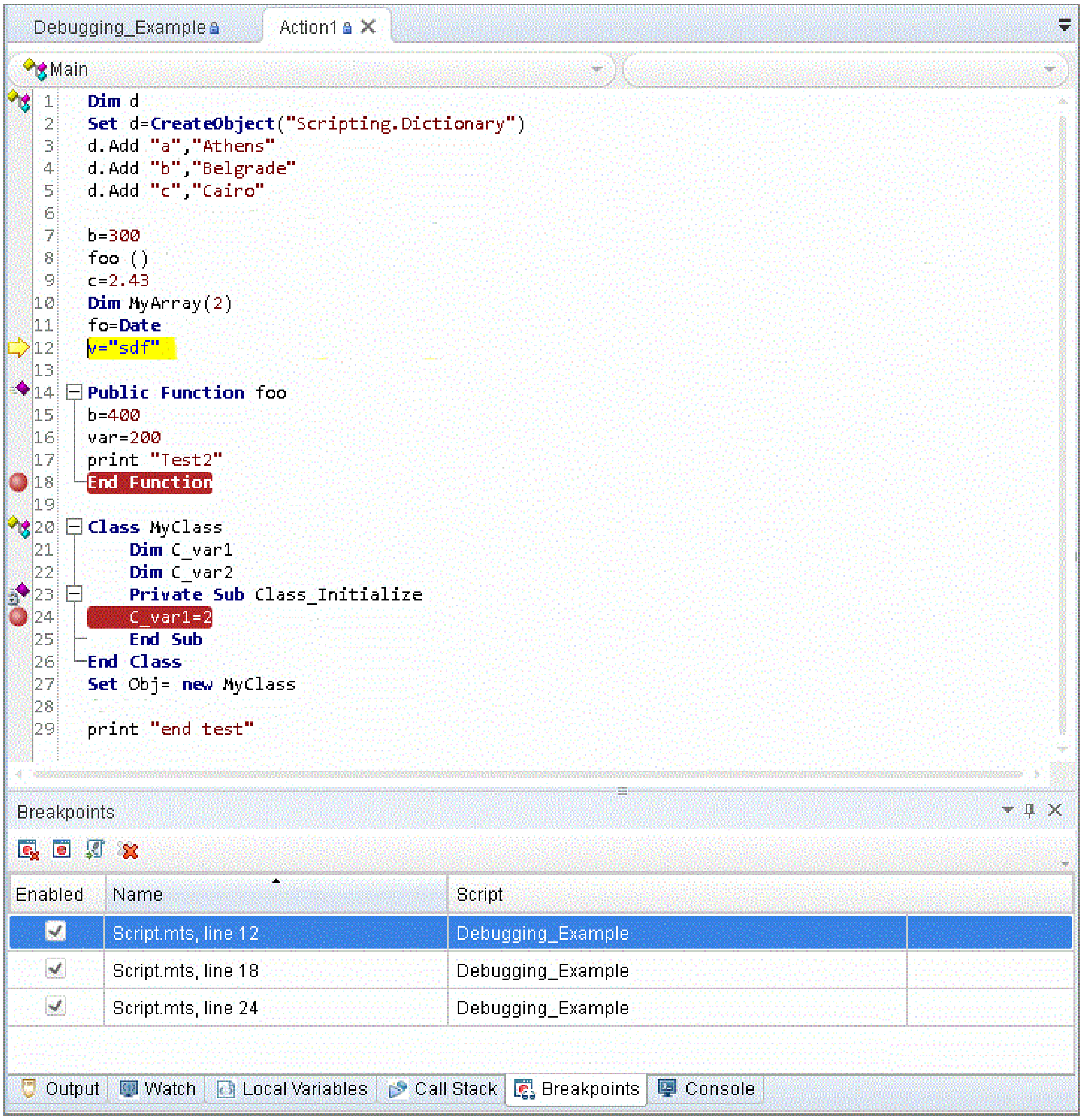
Debug Pane Interface

Relevant for: GUI actions, scripted GUI components, function libraries and user code files

The Debug panes enable you to perform different debugging activities when a run session is suspended, such as:

- View, set, or modify the current value of objects or variables in your test, scripted GUI component, function library, or user code file
- Run VBScript or C# commands in your paused run session
- Viewing information about the current call stacks of yourAPI test
- View information about the associated **.dll** files and threads being used in yourAPI run session

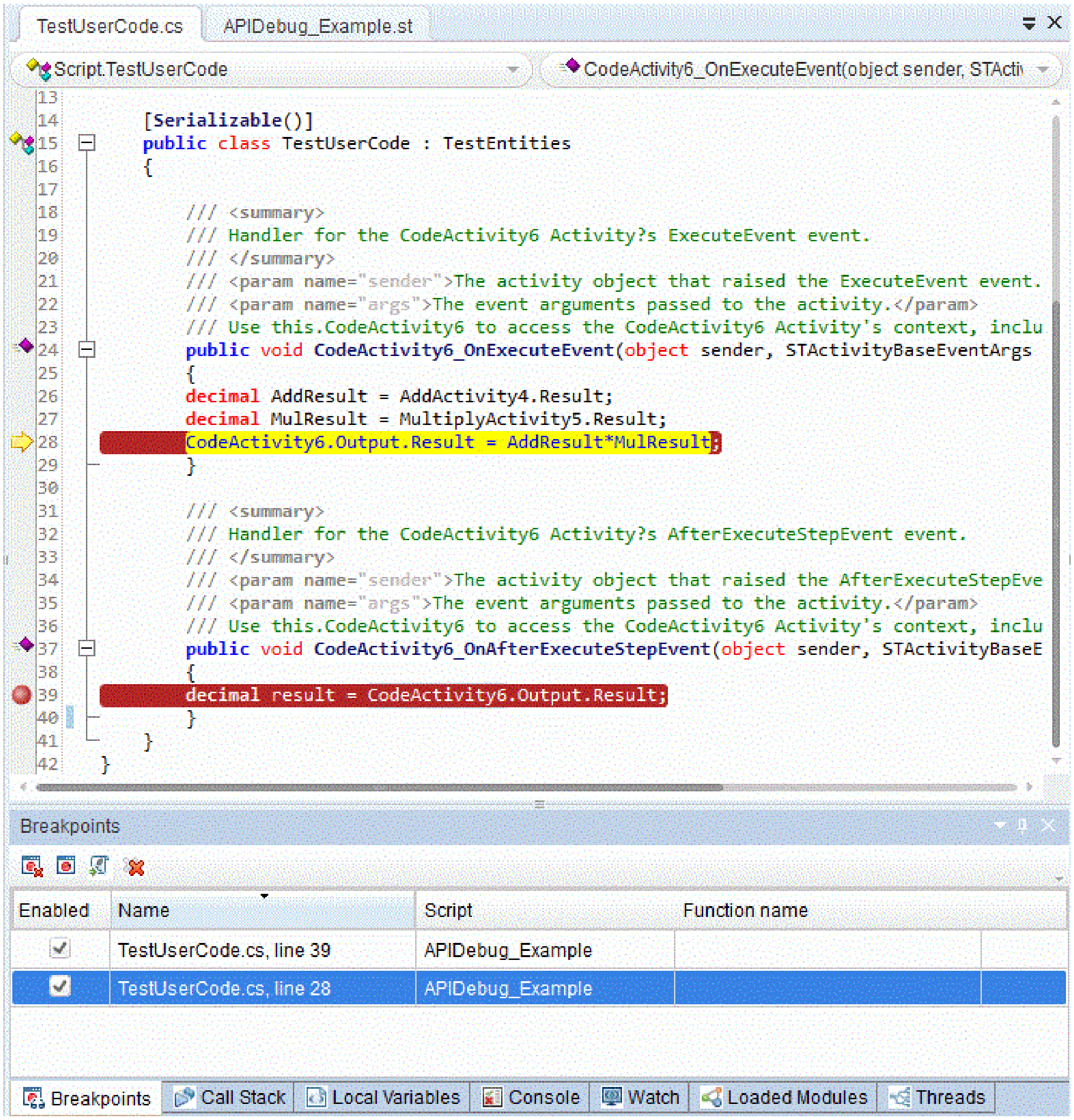
The image below shows all the Debug panes open as tabs for a **GUI test**. The default debug pane shown is the Debug Breakpoints tab.



```
1   Dim d
2   Set d=CreateObject("Scripting.Dictionary")
3   d.Add "a","Athens"
4   d.Add "b","Belgrade"
5   d.Add "c","Cairo"
6
7   b=300
8   foo ()
9   c=2.43
10  Dim MyArray(2)
11  fo=Date
12  v="sdf"
13
14  Public Function foo
15  b=400
16  var=200
17  print "Test2"
18  End Function
19
20  Class MyClass
21      Dim C_var1
22      Dim C_var2
23      Private Sub Class_Initialize
24      C_var1=2
25      End Sub
26  End Class
27  Set Obj= new MyClass
28
29  print "end test"
```

Enabled	Name	Script
✓	Script.mts, line 12	Debugging_Example
✓	Script.mts, line 18	Debugging_Example
✓	Script.mts, line 24	Debugging_Example

The image below shows all the Debug panes open as tabs for an **API event handler**. The default debug pane shown is the Debug Breakpoints tab.



```
13
14      [Serializable()]
15      public class TestUserCode : TestEntities
16      {
17
18          /// <summary>
19          /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
20          /// </summary>
21          /// <param name="sender">The activity object that raised the ExecuteEvent event.
22          /// <param name="args">The event arguments passed to the activity.</param>
23          /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
24          public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
25          {
26          decimal AddResult = AddActivity4.Result;
27          decimal MulResult = MultiplyActivity5.Result;
28          CodeActivity6.Output.Result = AddResult*MulResult;
29          }
30
31          /// <summary>
32          /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
33          /// </summary>
34          /// <param name="sender">The activity object that raised the AfterExecuteStepEve
35          /// <param name="args">The event arguments passed to the activity.</param>
36          /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
37          public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
38          {
39          decimal result = CodeActivity6.Output.Result;
40          }
41      }
42  }
```

Enabled	Name	Script	Function name
☑	TestUserCode.cs, line 39	APIDebug_Example	
☑	TestUserCode.cs, line 28	APIDebug_Example	

To access	1. Ensure that a test, component, function library, or user code file is in focus in the document pane. 2. Select **View > Debug** and choose the relevant debug pane.

Important information	Debug panes display information only when a run session is paused. A run session can be suspended in the following situations: - The run session stops at a breakpoint. - You use **Run** menu commands or context-menu options (such as **Pause**or **Run to Step**) to suspend a run session. - A step fails and you select the **Debug** option in the "Run Error Message Box" (described on page 701).
Relevant tasks	"How to Debug Your Test, Component, Function Library, or User Code File" on page 683
See also	- "Debugging Overview" on page 675 - "Considerations for Debugging GUI Tests and Components" on page 676 - "Considerations for Debugging API User Code Files" on page 676 - "How to Debug a GUI Action or a Function - Exercise" on page 689 - "How to Debug an API User Code File - Exercise" on page 693

The following debug panes are available:

UI Element	Description
Breakpoints pane	Displays the information about all breakpoints inserted into your action, scripted component, function library, or user code file and enables you to enable or disable any or all breakpoints in a run session. For details, see " Breakpoints Pane" on the next page.
Call Stack pane	Displays information about the functions, methods, or context currently relevant to your run session. For details, see "Call Stack Pane" on page 281.
Loaded Modules pane (API testing only)	Displays information about the **.dll** files associated with your run session and provides the path to find and debug them. For details, see "Loaded Modules Pane (API Testing) " on page 285.
Threads pane (API testing only)	Displays information about the threads running in your current run session. For details, see "Threads Pane (API Testing) " on page 287.

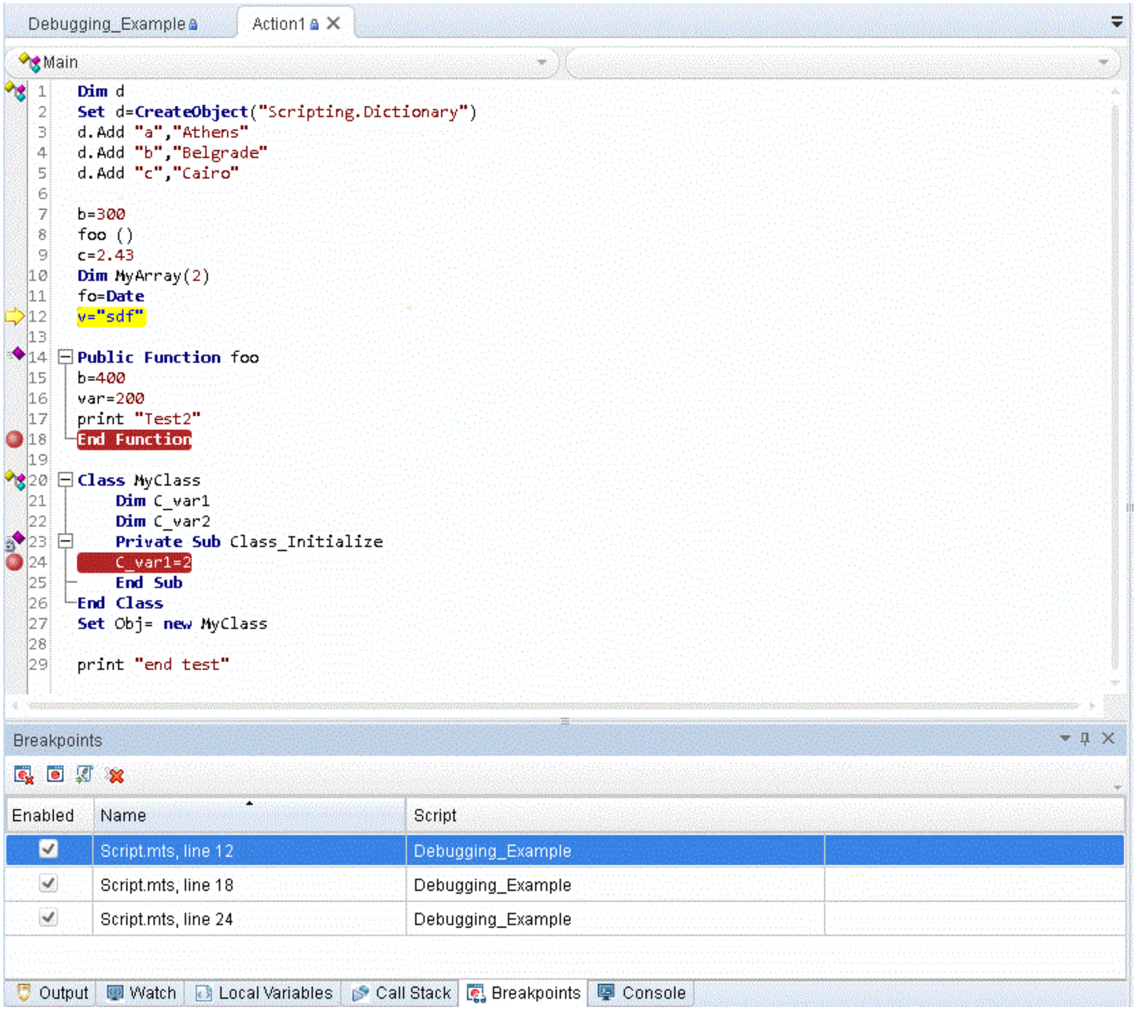
UI Element	Description
Local Variables pane	Displays the current values and types of all variables in the current context of the document. For details, see "Local Variables Pane" on page 288.
Console pane	Enables you to run VBScript or C# commands in your paused run session. For details, see "Console Pane" on page 292.
Watch pane	Displays the current values and types of variables and VBScript or C# expressions that you add to the Watch pane. For details, see "Watch Pane" on page 297.

Breakpoints Pane

Relevant for: GUI actions, scripted GUI components, function libraries and user code files

This debug pane enables you to view information about breakpoints inserted into yourGUI actions, scripted GUI components, function libraries or user code files and navigate directly to the breakpoint location in the relevant document.

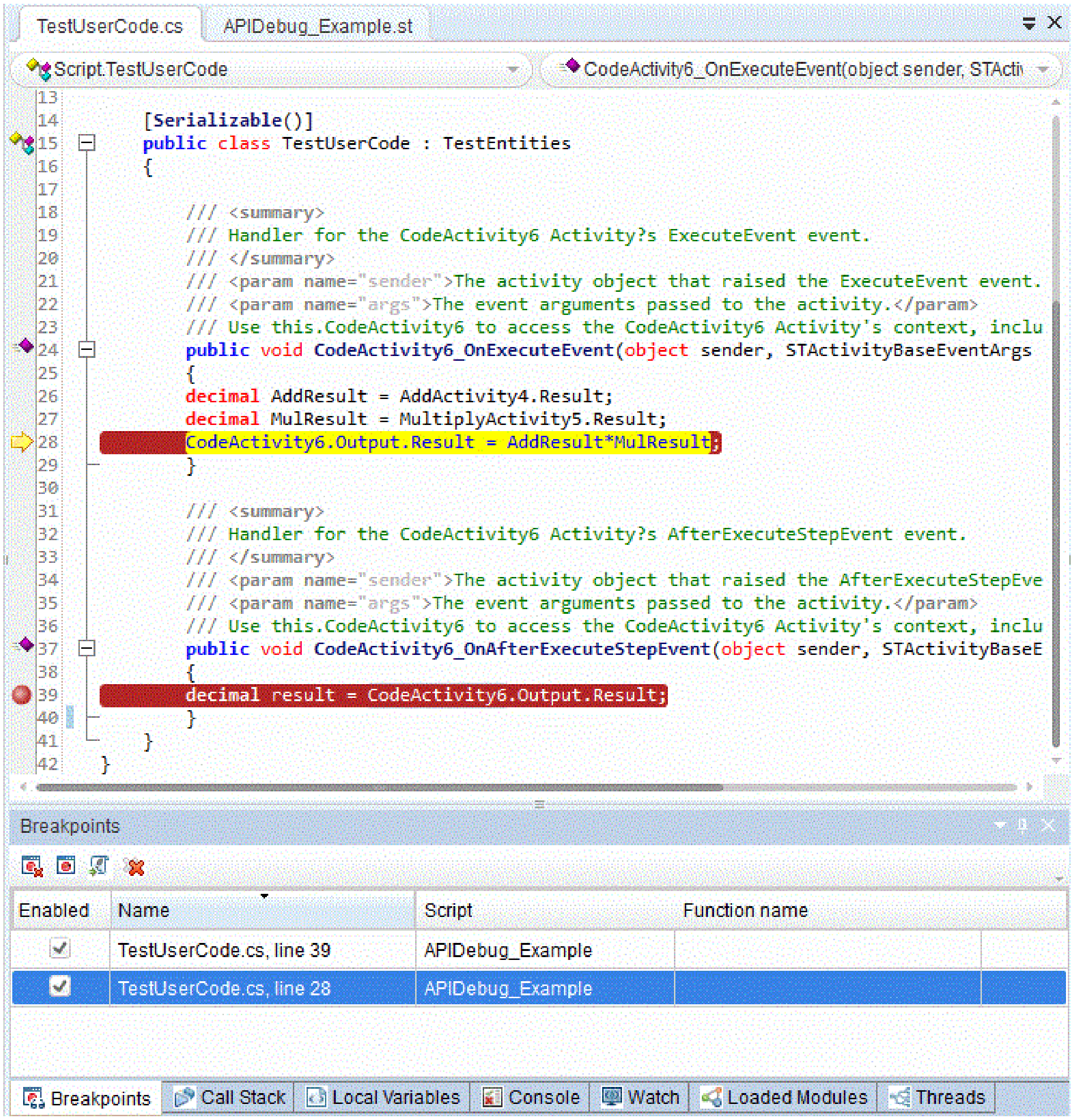
The image below shows the breakpoints pane for a **GUI test:**



```
1   Dim d
2   Set d=CreateObject("Scripting.Dictionary")
3   d.Add "a","Athens"
4   d.Add "b","Belgrade"
5   d.Add "c","Cairo"
6
7   b=300
8   foo ()
9   c=2.43
10  Dim MyArray(2)
11  fo=Date
12  v="sdf"
13
14  Public Function foo
15  b=400
16  var=200
17  print "Test2"
18  End Function
19
20  Class MyClass
21      Dim C_var1
22      Dim C_var2
23      Private Sub Class_Initialize
24      C_var1=2
25      End Sub
26  End Class
27  Set Obj= new MyClass
28
29  print "end test"
```

Enabled	Name	Script	
☑	Script.mts, line 12	Debugging_Example	
☑	Script.mts, line 18	Debugging_Example	
☑	Script.mts, line 24	Debugging_Example	

The image below shows the canvas for a **API test event handler.**



```

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

Enabled	Name	Script	Function name
☑	TestUserCode.cs, line 39	APIDebug_Example	
☑	TestUserCode.cs, line 28	APIDebug_Example	

To access	1. Do one of the following: ▪ Ensure that a test, component, function library, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, function library, or user code node. 2. Select **View > Debug > Breakpoints** **Note:** This pane displays information only when a breakpoint is inserted into a document.
Important information	• The pane displays the breakpoints for theGUI actions, scripted GUI components, function libraries, or API tests that are part of the current solution. • You can double-click on a breakpoint to navigate directly to the relevant line. • Breakpoints are saved with your test, scripted component, function library, or user code file and are maintained even after you close UFT.
Relevant tasks	"How to Use Breakpoints " on page 687
See also	"Breakpoints " on page 681

User interface elements are described below:

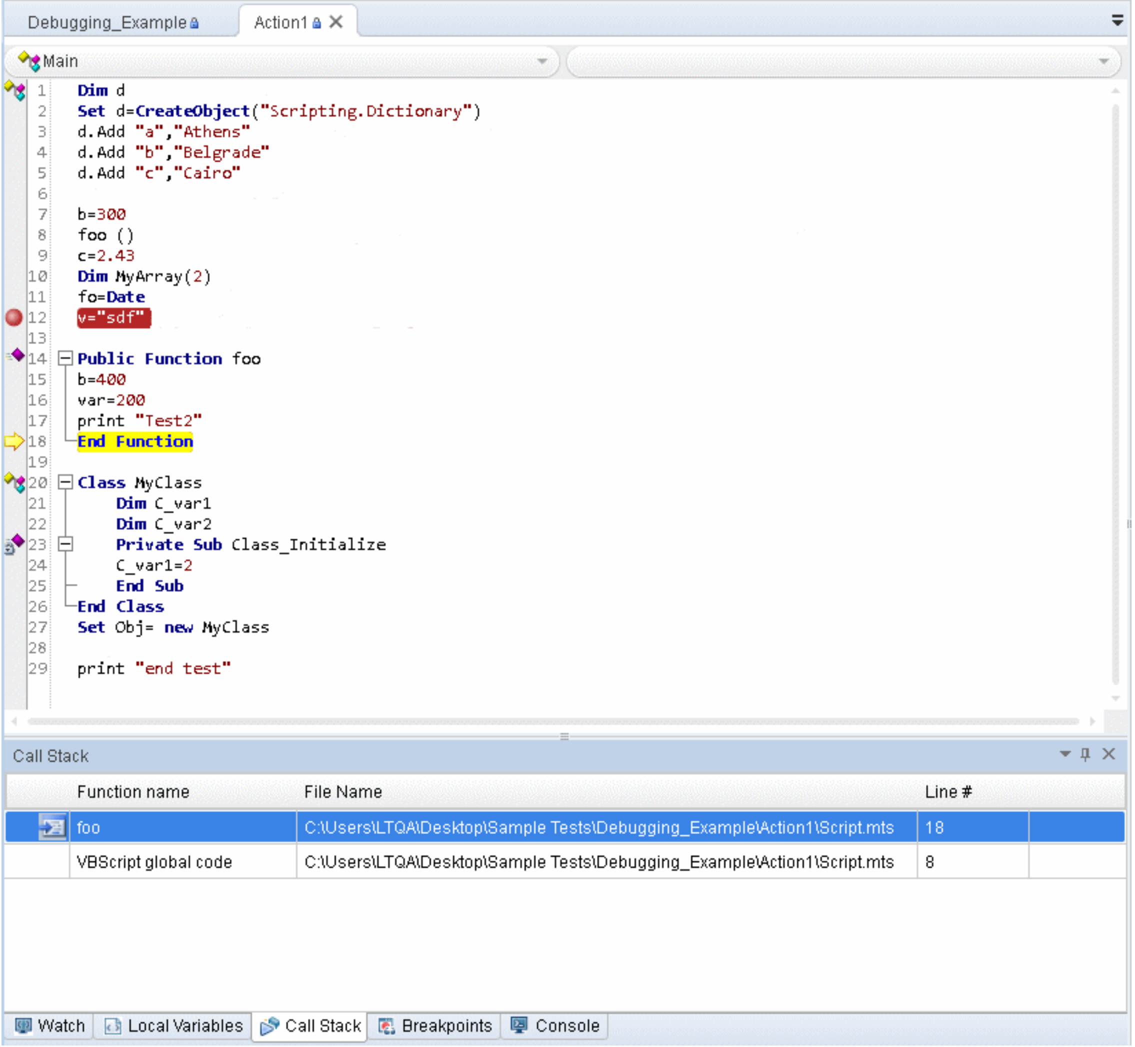
UI Element	Description
	Remove Current Breakpoint. Removes the current breakpoint from your document.
	Disable/Enable Breakpoint. Disables or enables the selected breakpoint, which instructs UFT to use or ignore it during a run session.
	Go to Source. Navigates directly to the line containing the selected breakpoint in your document.
	Remove All. Removes all breakpoints listed in the pane from the documents that contain them.
Enabled	A check box that specifies whether the breakpoint is enabled or disabled.
Name	The file name of the document that contains the breakpoint, and the number of the line in the file that contains the breakpoint.
Script	The name of the GUI action, scripted GUI component, function library, or user code file that contains the breakpoint.

Call Stack Pane

Relevant for: GUI actions, scripted GUI components, function libraries and user code files

This debug pane enables you to view information about the methods and functions that are currently on the call stack of your test, component, function library, or user code file or the context in which the run session was paused.

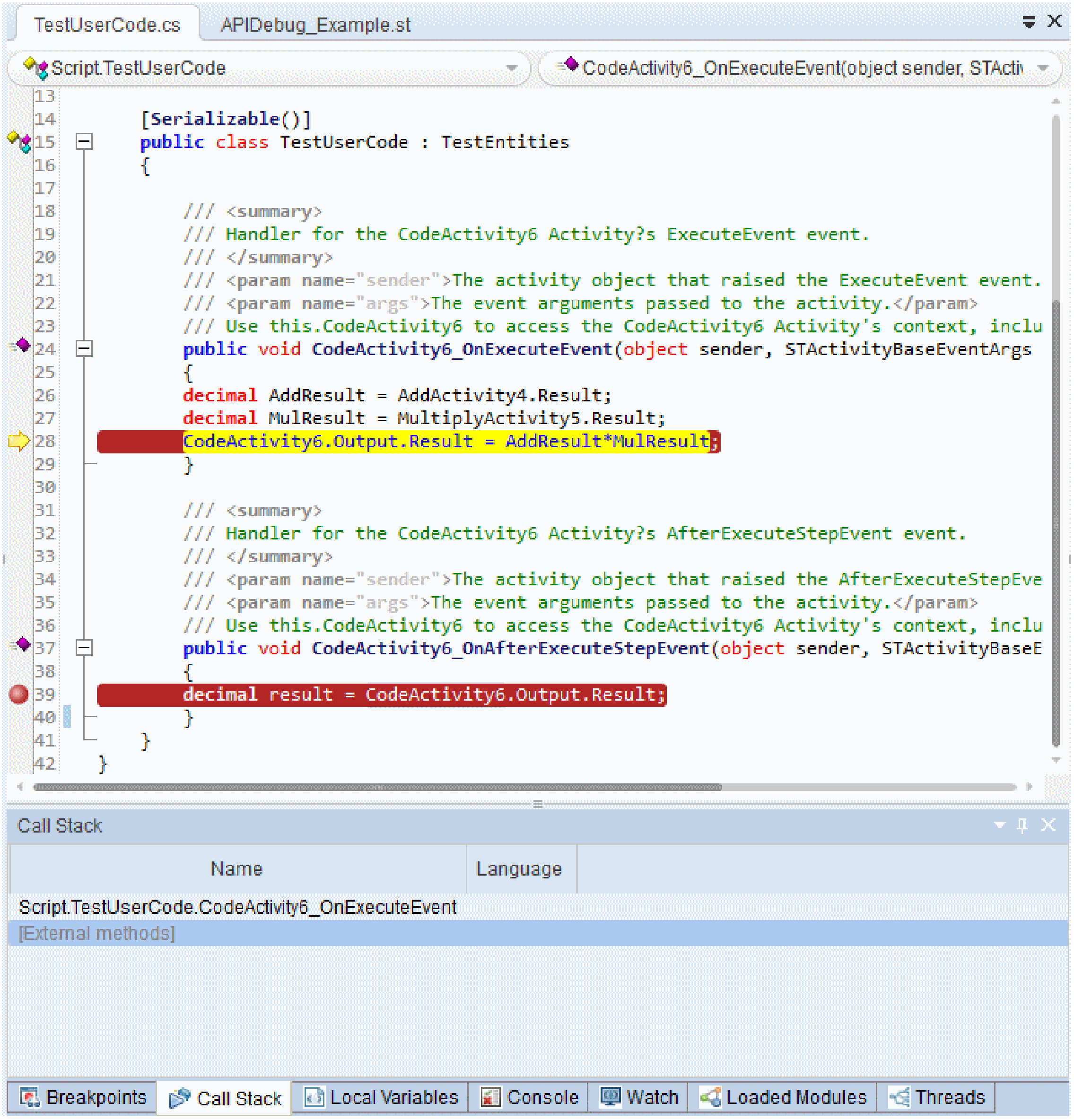
The image below shows the Call Stack pane for a **GUI test**.



```
Dim d
Set d=CreateObject("Scripting.Dictionary")
d.Add "a","Athens"
d.Add "b","Belgrade"
d.Add "c","Cairo"

b=300
foo ()
c=2.43
Dim MyArray(2)
fo=Date
v="sdf"

Public Function foo
b=400
var=200
print "Test2"
End Function

Class MyClass
    Dim C_var1
    Dim C_var2
    Private Sub Class_Initialize
    C_var1=2
    End Sub
End Class
Set Obj= new MyClass

print "end test"
```

	Function name	File Name	Line #
	foo	C:\Users\LTQA\Desktop\Sample Tests\Debugging_Example\Action1\Script.mts	18
	VBScript global code	C:\Users\LTQA\Desktop\Sample Tests\Debugging_Example\Action1\Script.mts	8

The image below shows the Call Stack pane for a **API testing event handler**.



```
    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

Name	Language
Script.TestUserCode.CodeActivity6_OnExecuteEvent	
[External methods]	

To access	1. Do one of the following: ▪ Ensure that a test, component, function library, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, function library, or user code node. 2. Select **View > Debug > Call Stack**. **Note:** This pane displays information only when a run session is paused.
Important information	• This pane is read-only. • You can navigate directly to the beginning line of a function, method, or context in your document by clicking on its row in the pane. If the relevant document is not open, UFT opens it. • The Watch pane and Local Variables pane display information relevant to the context that you select in the Call Stack pane. For example, if you select the row for a specific function in the Call Stack pane, the variables from this function and expressions selected to watch from this function are displayed in their respective panes. • **Caution for API testing users:** If you navigate to a stack that is not defined in a user code file, UFT opens other coding files. Do not edit this code. Doing so can cause unexpected behavior in your run sessions.

User interface elements are described below:

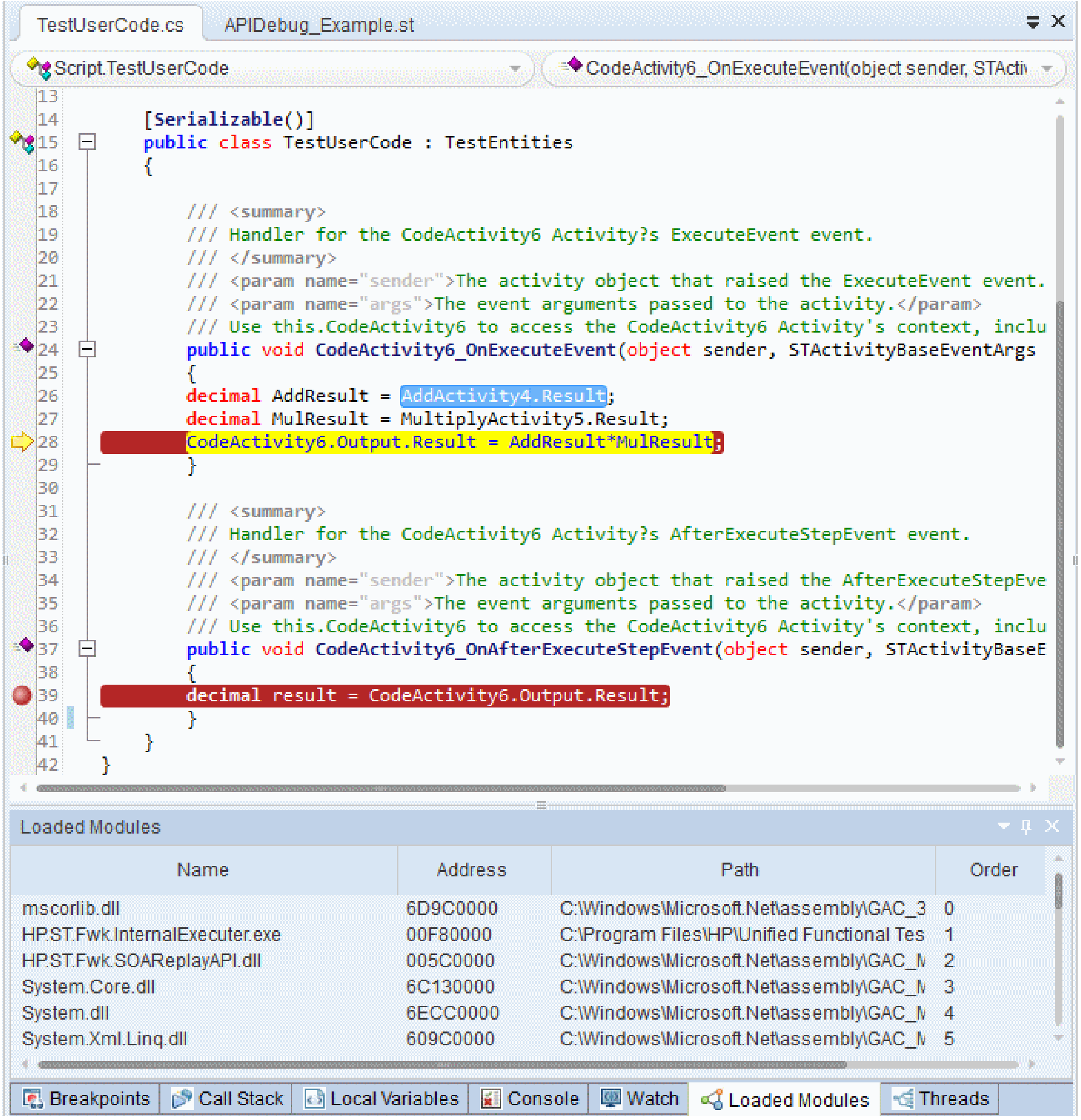
UI Element	Description
Function name	The name of the function, method, or context currently relevant. • If a run session is suspended before running a step or event or there is no specific function or event being run, then this pane displays the string VBScript global code (for GUI tests) or Script.TestUserCode.<event name> (for API tests). The expressions displayed in the Local Variables and Watch panes are evaluated within the context of the suspended step or event. • **For GUI testing:** If a run session is suspended within a function library, this pane initially displays the name of the function in which the run paused and enables you to switch to the context of other functions and subroutines within the same function library.

UI Element	Description
File name	The name of the file containing the called function, method, or context. This is the location in the file system or an ALM project.
Line #	The line number on which the function, method, or context definition begins. **Note: (for API testing)** Line numbers can be displayed by right-clicking and selecting **Show line number**.
Context menu options (API testing only)	The following context menu options are available only for API tests: • **Show external methods.** Displays all methods running in the background during a run session. **Note:** The Call Stack pane displays the external methods for informational purposes. You cannot navigate to the stack displayed in these method calls. • **Show module names.** Displays the .dll file in which the method call is found. • **Show argument values.** • **Show line number.** Displays the line number on which the method call begins for accessible internal methods.

Loaded Modules Pane (API Testing)

Relevant for: User code files only

This debug pane enables you to view information about the **.dll** files loaded and executed as an API test runs.



```

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
            decimal AddResult = AddActivity4.Result;
            decimal MulResult = MultiplyActivity5.Result;
            CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
            decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

Name	Address	Path	Order
mscorlib.dll	6D9C0000	C:\Windows\Microsoft.Net\assembly\GAC_3	0
HP.ST.Fwk.InternalExecuter.exe	00F80000	C:\Program Files\HP\Unified Functional Tes	1
HP.ST.Fwk.SOAReplayAPI.dll	005C0000	C:\Windows\Microsoft.Net\assembly\GAC_M	2
System.Core.dll	6C130000	C:\Windows\Microsoft.Net\assembly\GAC_M	3
System.dll	6ECC0000	C:\Windows\Microsoft.Net\assembly\GAC_M	4
System.Xml.Linq.dll	609C0000	C:\Windows\Microsoft.Net\assembly\GAC_M	5

To access	1. Do one of the following: ▪ Ensure that an API test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select an API test, component, or user code node. 2. Select **View > Debug > Loaded Modules** **Note:** This pane displays information only when a run session is paused.
Important information	The list of .dll files is displayed only when you pause a run session.

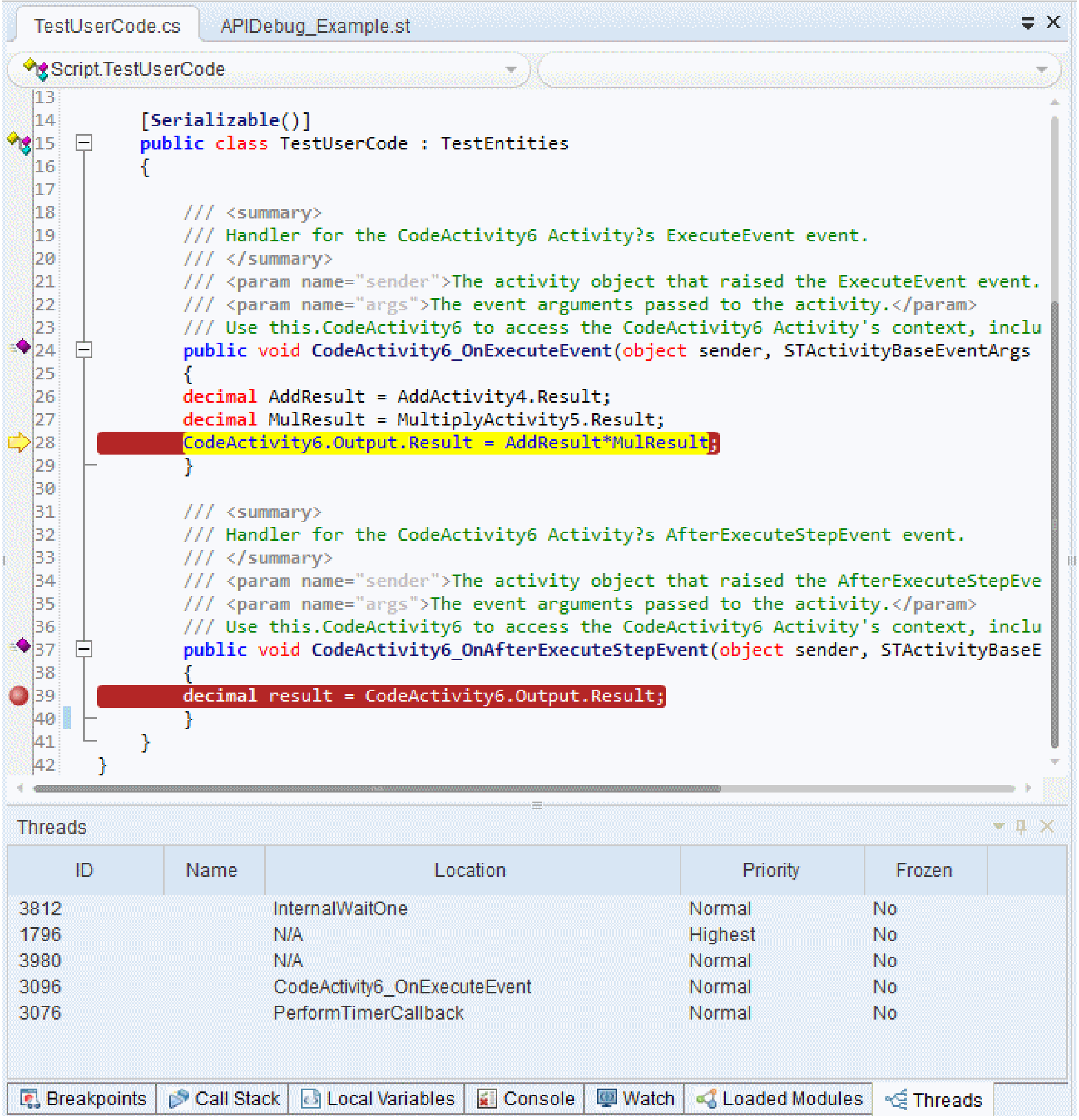
User interface elements are described below:

UI Element	Description
Name	The name of the .dll file.
Address	The numeric address of the .dll file.
Path	The location of the .dll file in the file system.
Order	The order in which the .dll files are used in your run session.
Symbols	The symbols included with each .dll file, which allow for debugging of loaded modules. **Note:** • A .dll file's symbols must be loaded to enable you to debug a .dll file. • Many of the .dll files listed in this pane have no symbols loaded, as they are files called by UFT during an API run session. • Any .dll files you add to your test can be debugged if there is a run error.

Threads Pane (API Testing)

Relevant for: User code files only

This pane enables you to view information about the threads currently running as part of the run session.



```
    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

ID	Name	Location	Priority	Frozen
3812		InternalWaitOne	Normal	No
1796		N/A	Highest	No
3980		N/A	Normal	No
3096		CodeActivity6_OnExecuteEvent	Normal	No
3076		PerformTimerCallback	Normal	No

To access	1. Do one of the following: ▪ Ensure that an API test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select an API test, component, or user code node. 2. Select **View > Debug > Threads** **Note:** This pane displays information only when a run session is paused.
Important information	• The list of threads is displayed only when you pause a run session. • If your test contains only one thread, then this pane is not available. • If the thread is contained within a user code file, you can double-click on the thread to navigate directly to the beginning of the thread.

User interface elements are described below:

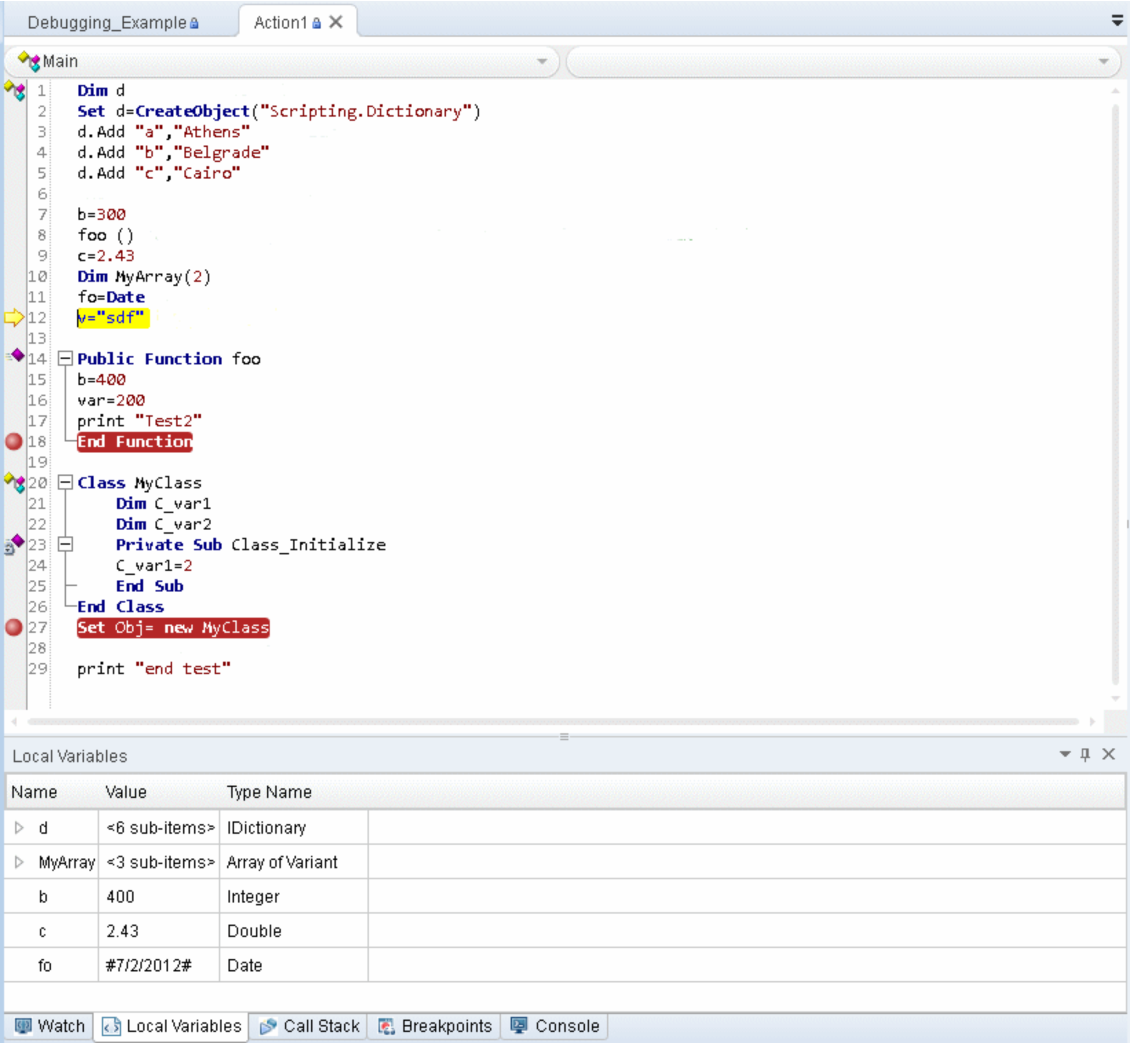
UI Element	Description
ID	The numeric identification of the thread.
Name	The name given to a particular thread.
Location	The location in which the thread is found. **Note:** Some threads are not found in your user code, but represent background threads that are part of your test. These threads are not accessible.
Priority	The order in which the threads can be used when running a multi-thread operation. Higher priority threads are called first.
Frozen	Indicates whether a thread is frozen or unfrozen. Right-click a thread to freeze or unfreeze it.

Local Variables Pane

Relevant for: GUI actions, scripted GUI components, function libraries and user code files

This debug pane displays the current values and types of all variables in current the context of your document.

The image below shows the Local Variables pane for a **GUI test**.

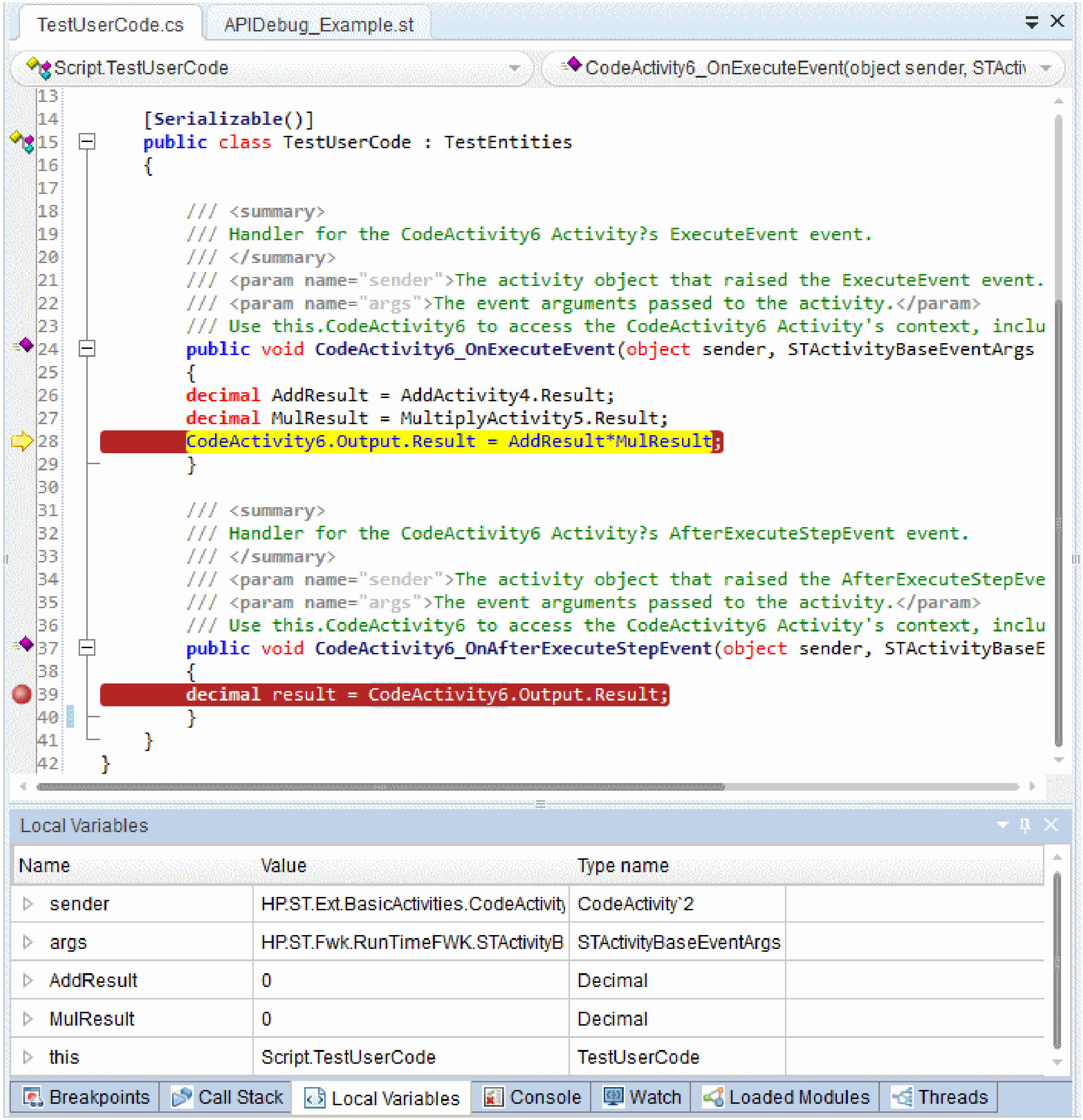


```
Dim d
Set d=CreateObject("Scripting.Dictionary")
d.Add "a","Athens"
d.Add "b","Belgrade"
d.Add "c","Cairo"

b=300
foo ()
c=2.43
Dim MyArray(2)
fo=Date
v="sdf"

Public Function foo
b=400
var=200
print "Test2"
End Function

Class MyClass
    Dim C_var1
    Dim C_var2
    Private Sub Class_Initialize
    C_var1=2
    End Sub
End Class
Set Obj= new MyClass

print "end test"
```

Local Variables

Name	Value	Type Name
▷ d	<6 sub-items>	IDictionary
▷ MyArray	<3 sub-items>	Array of Variant
b	400	Integer
c	2.43	Double
fo	#7/2/2012#	Date

The image below shows the Local Variables pane for a **API testing event handler**.



```

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

Name	Value	Type name
sender	HP.ST.Ext.BasicActivities.CodeActivity	CodeActivity`2
args	HP.ST.Fwk.RunTimeFWK.STActivityB	STActivityBaseEventArgs
AddResult	0	Decimal
MulResult	0	Decimal
this	Script.TestUserCode	TestUserCode

To access	1. Do one of the following: ▪ Ensure that a test, component, function library, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, function library, or user code node. 2. Select **View > Debug > Local Variables** **Note:** This pane displays information only when a run session is paused.
Important information	• The information displayed in this pane is read-only. • You cannot edit the value of expressions in the pane. To change the value of an expression, enter a command in the "Console Pane". • Only variables that were recognized up to the last step or event that occurred are displayed in the Local Variables pane. As you continue stepping through the subsequent steps in your document, UFT adds any additional variables that it recognizes and updates the values displayed in the Local Variables pane.
Relevant tasks	"How to Debug Your Test, Component, Function Library, or User Code File" on page 683
See also	• "Debugging Overview" on page 675 • "Considerations for Debugging GUI Tests and Components" on page 676 • "Considerations for Debugging API User Code Files" on page 676 • "Watching the Values of Variables and Properties of Objects During a Run Session" on page 680 • "How to Debug a GUI Action or a Function - Exercise" on page 689 • "How to Debug an API User Code File - Exercise" on page 693

User interface elements are described below:

UI Element	Description
Name	The name of the variable, or an expandable node whose child elements contain details about the variable. For example, if a variable is assigned to an object, you can expand the variable in the pane and view its methods and properties.

UI Element	Description
Value	The current value of the variable. **Note:** If a variable node contains child elements, the value column indicates the number of child elements.
Type Name	The type of the variable's value (for example, **Integer** or **String**).

Console Pane

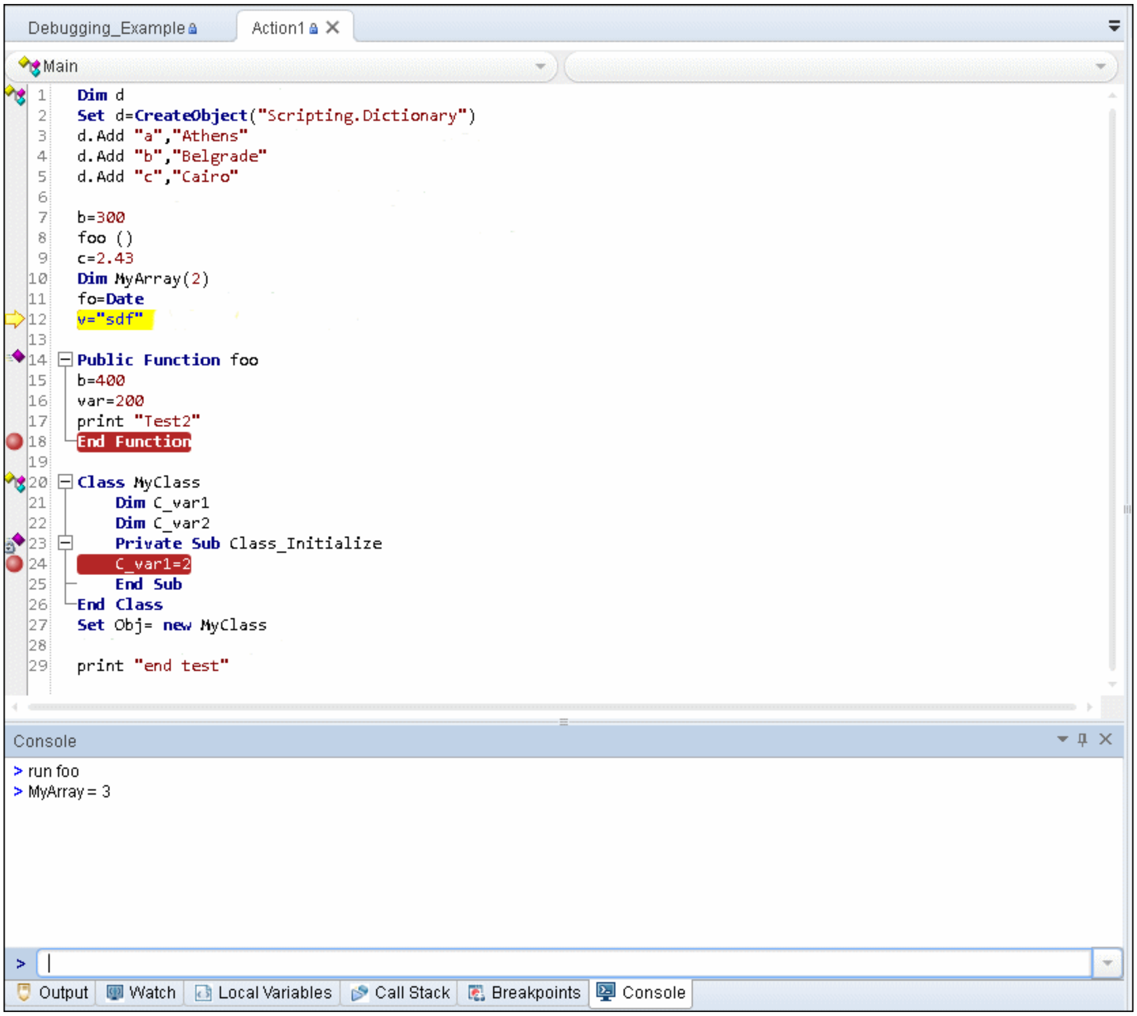
Relevant for: GUI actions, scripted GUI components, function libraries and user code files

This debug pane enables you to run lines of VBScript code or C# code in your suspended run session.

For example, you can run code that performs any of the following activities before you resume the run session:

- Retrieves information from the application you are testing
- Runs a test object method and displays the return value, enabling you to learn more about how the method works
- Modifies the value of a native (run-time object) property in the application
- Calls a native (run-time object) method in the application
- Modifies the input and output properties for a step
- Sets or modifies a variable

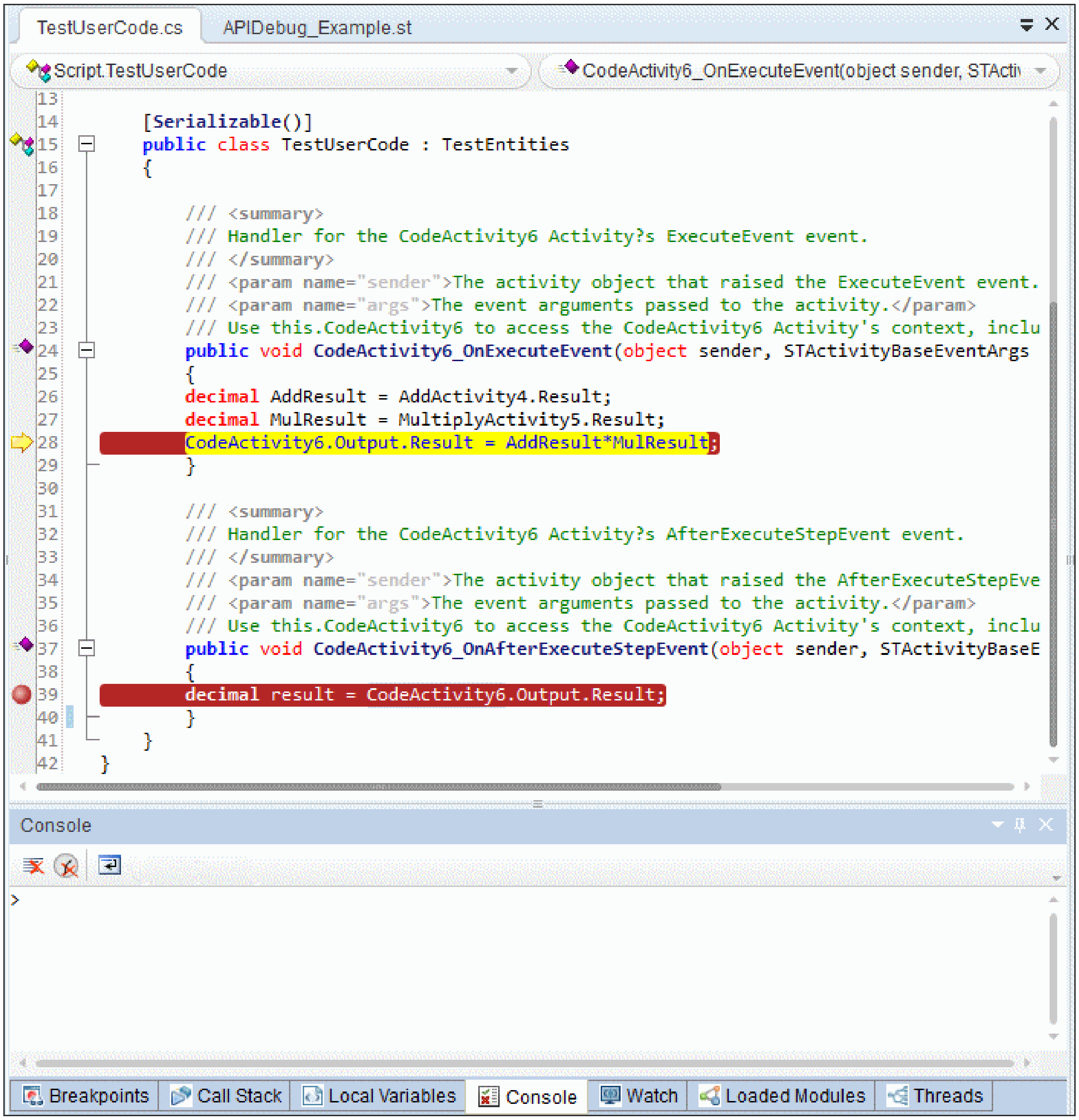
The image below shows the Console pane for a **GUI test.**



```
Dim d
Set d=CreateObject("Scripting.Dictionary")
d.Add "a","Athens"
d.Add "b","Belgrade"
d.Add "c","Cairo"

b=300
foo ()
c=2.43
Dim MyArray(2)
fo=Date
v="sdf"

Public Function foo
b=400
var=200
print "Test2"
End Function

Class MyClass
    Dim C_var1
    Dim C_var2
    Private Sub Class_Initialize
    C_var1=2
    End Sub
End Class
Set Obj= new MyClass

print "end test"
```

```
> run foo
> MyArray = 3
```

The image below shows the Console pane for a **API testing event handler.**



```

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

```
>
```

To access	1. Do one of the following: ▪ Ensure that a test, component, function library, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, function library, or user code node. 2. Select **View > Debug > Console** **Note:** This pane displays information only when a run session is paused.
Important information	• You can enter lines of code in the Console pane only when a run session is suspended. When no run session is suspended, you can view the command history, select and copy text from it, or use the **Clear All** context menu. • **For GUI testing:** Running **RunAction** or **LoadAndRunAction** statements from this pane is not supported.
Relevant tasks	"How to Debug Your Test, Component, Function Library, or User Code File" on page 683
See also	• "Debugging Overview" on page 675 • "Considerations for Debugging GUI Tests and Components" on page 676 • "Considerations for Debugging API User Code Files" on page 676 • "Watching the Values of Variables and Properties of Objects During a Run Session" on page 680 • "How to Debug a GUI Action or a Function - Exercise" on page 689 • "How to Debug an API User Code File - Exercise" on page 693

UI elements for GUI testing

Console pane user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<command line prompt>	Enables you to enter a line of code to be run in the context of your suspended run session. Type or paste the line of code in the window and press ENTER to run the code.

UI Element	Description
<command line history>	Displays the lines of code that you ran. • You cannot make any changes to these lines, but you can select and copy text from them. • You can use the UP and DOWN arrow keys to browse through the command history. UFT copies the commands to the active command line, enabling you to repeat or reuse commands that you entered earlier.
<right-click context menu>	Provides commands that you can use to edit the content of the Console pane. • The **Cut**, **Copy**, and **Paste** commands enable you to use the clipboard to copy text from the command history and to edit the active command line. • The **Clear All** command enables you to erase all of the command history.

UI elements for API testing

Console pane user interface elements are described below (unlabeled elements are shown in angle brackets):

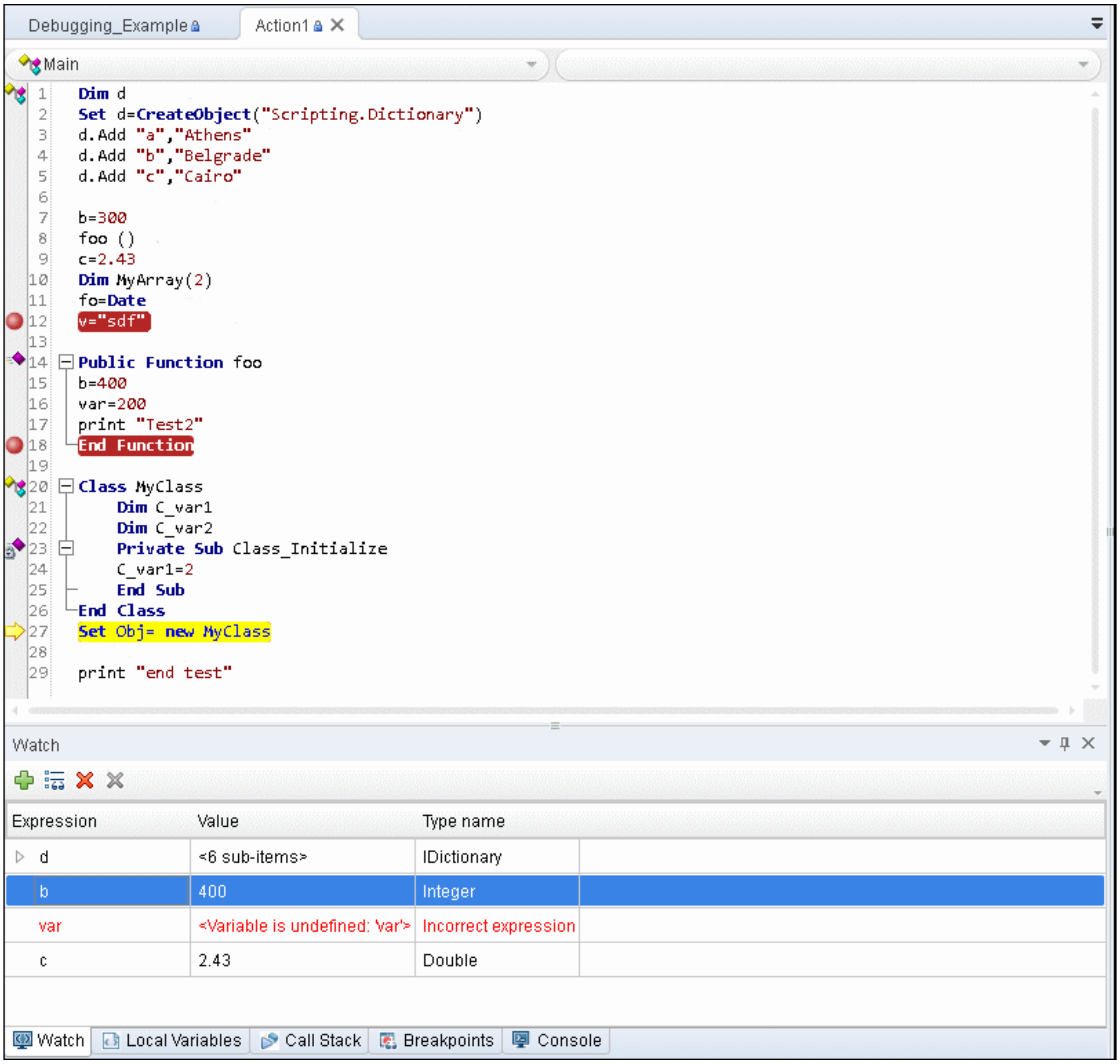
UI Element	Description
	Clear Console. Enables you to erase all of the commands entered in the command line area.
	Delete History. Clears all previously saved commands. By default, all previous commands are saved in the pane and can be selected for a new command by using the up or down arrows.
	Toggle Word Wrap. Enables or disables word wrap behavior for commands given in the command line area.
<command line area>	Enables you to enter a line of code to be run in the context of your suspended run session. Type or paste the line of code in the window and press ENTER to run the code. This area also displays the command line history, including the lines of code that you ran. **Notes:** • You cannot make any changes to these lines, but you can select and copy text from them. • You can use the UP and DOWN arrow keys to browse through the command history. UFT copies the commands to the active command line, enabling you to repeat or reuse commands that you entered earlier.

Watch Pane

Relevant for: GUI actions, scripted GUI components, function libraries and user code files

This debug pane enables you to view the current values and types of selected variables, properties, and VBScript or C# expressions in your suspended run session.

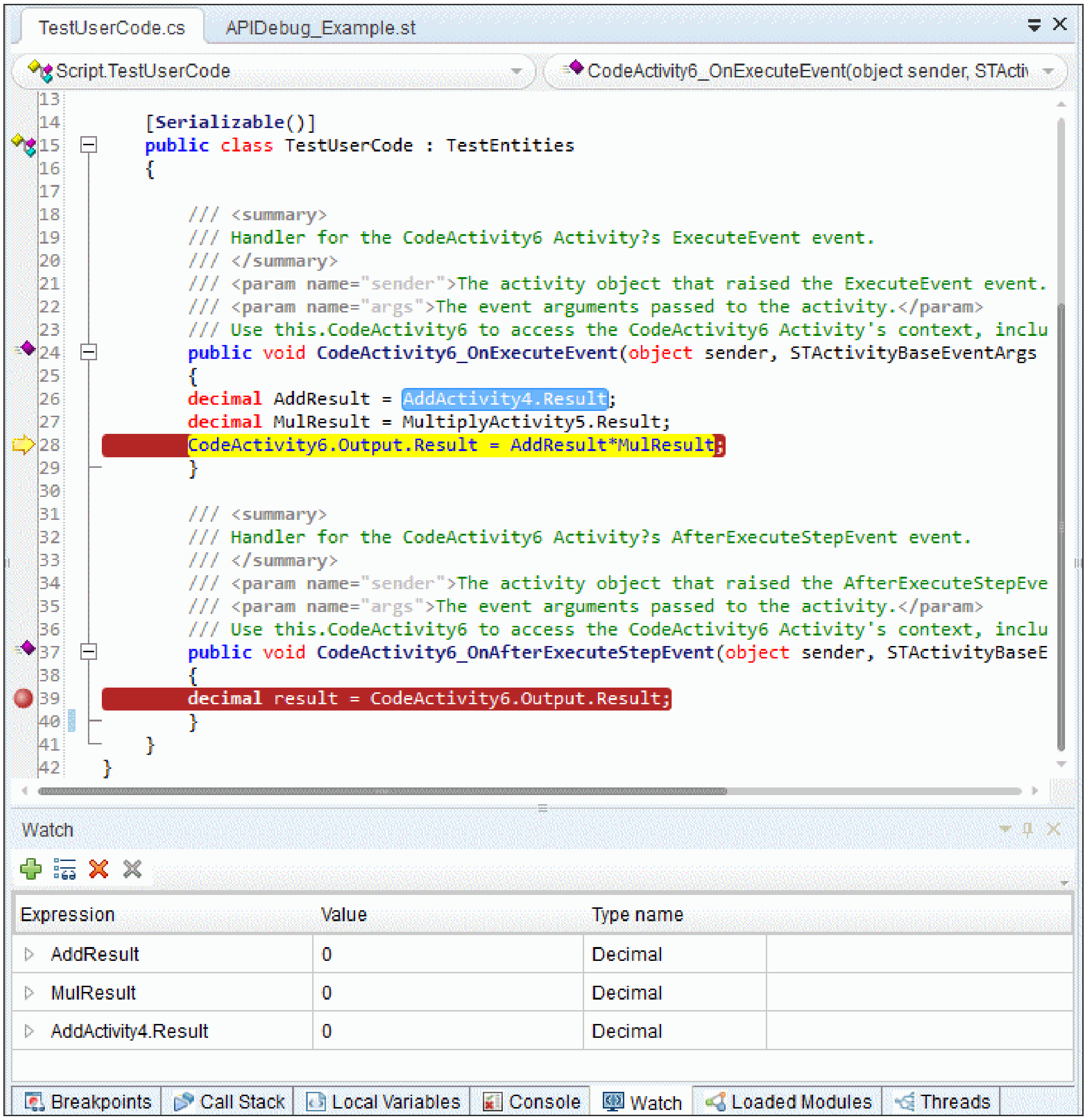
The image below shows the Watch pane for a GUI test.



```
Dim d
Set d=CreateObject("Scripting.Dictionary")
d.Add "a","Athens"
d.Add "b","Belgrade"
d.Add "c","Cairo"

b=300
foo ()
c=2.43
Dim MyArray(2)
fo=Date
v="sdf"

Public Function foo
b=400
var=200
print "Test2"
End Function

Class MyClass
    Dim C_var1
    Dim C_var2
    Private Sub Class_Initialize
    C_var1=2
    End Sub
End Class
Set Obj= new MyClass

print "end test"
```

Expression	Value	Type name
d	<6 sub-items>	IDictionary
b	400	Integer
var	<Variable is undefined: 'var'>	Incorrect expression
c	2.43	Double

The image below shows the Watch pane for an API testing event handler.



```
    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the ExecuteEvent event.
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEve
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, inclu
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseE
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

Expression	Value	Type name
AddResult	0	Decimal
MulResult	0	Decimal
AddActivity4.Result	0	Decimal

To access	1. Do one of the following: ▪ Ensure that a test, component, function library, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, function library, or user code node. 2. Select **View > Debug > Watch**. **Note:** This pane is only relevant when a run session is paused.

Important information	• You can add, edit, or remove expressions within the Watch pane using the pane's toolbar buttons in addition to the **Run** menu command (**Run > Add to Watch**). • You cannot edit the value of expressions in the pane. To change the value of a watched expression, enter a command in the "Console Pane". • You can expand Watch pane items if the property selected contains multiple values. • You can sort the columns in the watch pane by expression, value, or type name by clicking the column headers. • Watch expressions are saved with the test and maintained between testing sessions and as you switch between open documents. • **For API testing:** If you added an item to this pane from an user code file that already ran, the pane will show it as undefined.
Relevant tasks	"How to Debug Your Test, Component, Function Library, or User Code File" on page 683
See also	• "Debugging Overview" on page 675 • "Considerations for Debugging GUI Tests and Components" on page 676 • "Considerations for Debugging API User Code Files" on page 676 • "Watching the Values of Variables and Properties of Objects During a Run Session" on page 680 • "How to Debug a GUI Action or a Function - Exercise" on page 689 • "How to Debug an API User Code File - Exercise" on page 693

User interface elements are described below:

UI Element	Description
✚	**Add New Watch Expression.** Opens the Add New Watch dialog box, which enables you to add a new expression to the Watch pane. After you enter the expression and click **OK**, UFT displays the value and type of the expression in the pane.
☰	**Edit Watch Expression.** Enables you to edit the selected expression during your debug session.
✖	**Remove Watch Expression.** Removes the currently selected expression from the Watch pane.
✖	**Remove All.** Removes all the expressions currently displayed in the Watch pane.

UI Element	Description
Expression	The expression whose value you want to watch. For details on adding and removing expressions from the Watch pane, see "Watching the Values of Variables and Properties of Objects During a Run Session" on page 680. **Caution:** UFTruns the expressions in the Watch pane to evaluate them. Therefore, do not enter a test object method or any expression whose evaluation could affect the state of the test or a test object, as this can lead to unexpected behavior of your test or function library.
Value	The current value of the expression. The evaluated value is displayed only when a run session is suspended.
Type name	The type of the expression's value after it is evaluated (for example, **Integer** or **String**). If an expression cannot be evaluated in the current context, the type displayed is **Incorrect expression**.

Troubleshooting and Limitations - Debug Panes

Relevant for: GUI actions, scripted GUI components, function libraries and user code files

This section describes troubleshooting and limitations for the Debug Pane.

- You cannot edit variables or properties from the Local Variables or Watch Panes.

  Workaround: Use the Console pane to edit and modify your variable and property values by running a command line to assign the new value.

- **For GUI testing:** (Relevant only if Microsoft PDM 9.x or later is installed on your computer.) If you add Action automation objects to the Watch pane and then close and re-open your test without closing and re-opening UFT, these actions may not load successfully in the re-opened test.

  Workaround: Restart UFT and open your test.

Chapter 10: Document Pane

Relevant for: GUI tests and components, API testing, and business process testing

This chapter includes:

Concepts

Document Pane Overview

Relevant for: GUI tests and components, API testing, and business process testing

The document pane is the main design area in UFT and displays all open documents as separate tabs. Each tab can also be individually moved, docked, or floated as an independent pane, and you can drag and drop or pin any of the UFT panes within the document pane.

For details on how to move UFT panes, see "How to Customize the UFT Window " on page 184. You can also restore the tabs or panes to the default location from the **View** menu.

The document pane displays the following types of documents:

GUI testing documents	• GUI tests • GUI actions • Function libraries
API testing documents	• API tests • API actions • C# files containing event handler code (TestUserCode.cs files) • C# user code files
BPT documents	• Business process tests • Business process flows • Business components, including API, keyword GUI, and scripted GUI components • Application areas
Non-testing documents	• Text files

The features available for viewing and editing documents depend on the document type, as described in the following table.

Document Type	Opens in the:
• GUI and API tests • API actions • API business components	**Canvas.** Graphically displays and enables you to edit the flow of your test, action, or component. For details, see: "The Canvas" on page 209
• GUI actions • Scripted GUI components • Keyword GUI components	**Keyword View.** Enables you to create and view the steps of your action or component in a keyword-driven, modular, table format. For details: "Keyword View User Interface" on page 942
• GUI actions • Scripted GUI components • Function libraries • C# code files • Text files	**Editor.** Provides text and code editing features that facilitate the design of your text, script, and code documents. For details: • **General Editor information:** "Editing Text and Code Documents" on the next page • **For GUI testing:** "Programming in GUI Testing Documents in the Editor - Overview " on page 995 • **For API testing:** "How to Open a Window for Writing Custom Code" on page 1948
• Business process tests • Business process flows	**BPT in UFT**. Displays and enables you to edit business process tests and flows, including grouping components and flows, and modifying run order. For details, see "Business Process Testing in UFT - Overview" on page 2065
Application areas	**Application Area.** Displays and enables you to edit the application area settings and resource associations. For details: "Application Area User Interface" on page 2101

Multiple Document Types in the Document Pane

Relevant for: GUI tests and components and API testing

Using the document pane, you can open and work with multiple documents simultaneously. To open multiple documents, they must all be included in the same solution.

> **Note:** You can open multiple function libraries independently from a solution.

You can navigate between multiple documents by clicking the tab name of the document you want to view. You can also use the document view drop-down arrow ▼ to display a list of open documents.

For details on managing documents, see "How to Create and Manage Documents" on page 136.

Editing Text and Code Documents

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

You can write customized code for your tests by modifying actions, function libraries, and user code files in the Editor, as well as enhance your tests and function libraries using a few simple programming techniques.

The Editor supports common text and code editing features. For details, see:

- "Statement Completion in the Editor" on the next page
- "Automatic Code Completion" on page 316
- "Bookmarks Overview" on page 204
- "Searching and Replacing in the Editor " on page 317
- For GUI testing: "Programming in GUI Testing Documents in the Editor" on page 994
- For API testing: "Writing Event Handlers for API Test Steps" on page 1935

Other notable Editor features include expanding and collapsing code, zooming in and out using the mouse, and code templates defined in the Code Templates pane of the Options dialog box (**Tools > Options > Coding** tab **> Code Templates** node). For details about defining code templates, see "Code Templates Pane (Options Dialog Box > Coding Tab)" on page 573.

To define other preferences related to viewing and editing documents in the Editor, see "Text Editor Tab (Options Dialog Box)" on page 574.

For a user interface description of the Editor, see "Editor User Interface" on page 334.

Statement Completion in the Editor

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

Statement completion, which is similar to Microsoft's IntelliSense functionality, enables you to increase programming speed and accuracy by providing dynamic lists of items, in the form of tooltips, drop-down lists, or popup windows, while writing statements in the Editor.

As you type in the Editor,UFT displays items you might want to add to your statement, as well as the syntax relevant to what you are typing. UFT provides this type of statement completion information, when available, for:

- Activity-specific properties (API testing only)
- Function and method syntax
- Objects you create in your code
- Operations
- Properties or operations that return objects
- Structured parameters
- Variable definitions and methods
- Variables to which classes, objects or test objects are assigned
- VBScript classes, user defined functions, or constants (GUI actions and scripted components only)
- Reserved objects
- Test objects and collections (GUI actions and scripted components only)
- Utility objects

> **Note:** (**for GUI testing only)** The list of available statements is displayed even if you typed an object that has not yet been added to the object repository. If the action contains a function, or the action or component is associated with a function library, the functions are also displayed in the list.

For details about statement completion in the Editor, see:

- "Statement Completion Options" on the next page
- "Statement Completion Considerations" on page 311

- "Statement Completion Example for GUI Testing" on page 313
- "Troubleshooting and Limitations - Statement Completion" on page 363

Statement Completion Options

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

The following table summarizes some statement completion options available when you enter specific items and keystrokes.

If you enter:	Followed by:	UFT displays:
An operation name	SPACE or Open parenthesis"("	The operation syntax, including its mandatory and optional arguments. When you add a step that uses an operation, you must define a value for each mandatory argument associated with the operation.
An argument	Comma (,)	The operation syntax, bolding the next argument for which you need to enter a value. Relevant if you enter a comma after any argument value other than the last one in a step. **Note:** For certain operations, when you type the space or comma before an argument that has a predefined list of values, UFT displays the list of possible values.
An operation or function name	CTRL+SHIFT +SPACE or **For GUI testing:** Select **Edit > Format > Argument Info**	The statement completion (argument syntax) tooltip for that item.

If you enter:	**Followed by:**	**UFT displays:**
CTRL+SPACE or **For GUI testing:** Select **Edit > Format > Complete Word**		A dynamic list of the relevant: • operations • properties • user-defined functions • constants and local variables relevant to the current programming scope • functions and methods relevant to the current programming scope **GUI testing only:** • test objects • utility objects • collections **Note:** If there is only one relevant item defined, its name is automatically entered in the step, without opening the list. For example, if you typed the beginning of the item name before pressing CTRL+SPACE, and only one item matches the text you typed.
The beginning of an item in the statement completion list		The list of items, highlighting the first item (alphabetically) that matches the text you typed. Pressing ENTER or SPACE adds the highlighted word to the step.

Statement Completion Options for GUI Testing Only

The following table summarizes some additional statement completion options available in GUI tests only, when you enter specific items and keystrokes.

If you enter:	Followed by:	UFT displays:
• A test object type • "Environment" • "Parameter" **(for tests and scripted components)**	Open parenthesis"("	Respectively, an alphabetically sorted list of the relevant available: • Test objects (for example, **Page(** displays a list of all the Page test objects in the object repository) • Environment variables (built-in, user-defined, and external) • Action parameters (output and input) If there is only one selection available, UFT automatically enters its name in quotes after the open parenthesis.
• A test object • A collection object • A variable that was assigned an object • An object that you created in the test or component (using **CreateObject**, for example)	Period (.)	Depending on the type of object and its hierarchy, a dynamic list of the relevant: • test objects (Editor only) • collections • operations • properties • registered functions **Example:** • The list displayed for a container object includes the objects it contains. • The list displayed for a collection object includes collection methods. • The list displayed for a variable that was assigned an Excel worksheet (Set var = CreateObject ("excel.application")) includes worksheet methods and properties.

If you enter:	Followed by:	UFT displays:
A variable that was assigned a VBScript class	Period (.)	A list of the relevant VBScript class methods, properties, and variables
A **With** statement	Period (.)	A list of the operations and properties available for the relevant object. **Note:** If you manually type a **With** statement (as opposed to using a menu command to create it), you must use the **Edit > Format > Apply "With" to Script** command (or press CTRL+W) to enable statement completion within the **With** statement.
The **Object** property	Period (.)	A list of the object's native operations and properties. Available **only if** the object data is currently available in the Active Screen (tests and scripted components only) or the open application. For details related to tests and scripted components, see "Statement Completion Considerations" on the next page.
A structured parameter	Colon (:)	A list of the elements available, according to the structure definition. **Example:** If you define a ZMOVIE structure, with the elements TITLE, DIRECTOR, GENRE, and STARRING, and then map a parameter named Movie to a structure of this type, then when you type: Parameter ("Movie: the displayed list includes the elements of a ZMOVIE structure: TITLE, DIRECTOR, GENRE, STARRING.

If you enter:	Followed by:	UFT displays:
A structured parameter's element or sub-element	Period (.)	A list of the sub-elements available, according to the structure definition. **Example:** If you type: Parameter ("Movie:Starring.item[1]. the displayed list includes the sub-elements of a Starring element: FIRST_NAME, LAST_NAME.

Statement Completion Considerations

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

When working with UFT's statement completion feature, consider the following:

- To close the statement completion drop-down list without selecting from it, press ESC.
- If you resize the frame in which the statement completion drop-down list is displayed, UFT subsequently uses the new size when it displays statement completion drop-down lists.
- UFT might not display statement completion information if the statement is typed incorrectly and contains syntax errors. In many cases, you can view such errors in the "Errors Pane" (described on page 364) when you save your changes.

Statement Completion Considerations for GUI Testing Only

- Although statement completion in function library documents is supported to help generate test object statements, it is generally not recommended to include a full object hierarchy statement in a function. It is preferable to make your functions generic so that they can be used with different objects.
- In some cases, UFT needs to retrieve statement completion information from an object in the application. In such cases, you may experience a delay while typing in the Editor.
- When you record in a browser other than Internet Explorer, UFT captures the dynamic HTML from the relevant browser. However, the Active Screen uses an Internet Explorer engine and thus renders Web-based pages as static HTML with the Internet Explorer DOM exposed.

  Therefore, if you insert the **Object** property for a Web object when only Active Screen object data is available (the object is not displayed in an open application), then UFT statement completion displays the available native operations and properties for the Internet Explorer DOM, even if you recorded the object in another browser.

  For details on the **Object** property, see "Native Properties and Operations" on page 1008.

- In the Editor, when working with Java or ActiveX objects,UFT dynamically retrieves the list of possible values for certain arguments from the object in the application. For UFT to retrieve the possible values, the application must be open and the relevant object must be visible. For example, UFT can retrieve the list of items in a specific Java list object, and display them as the possible values for the Item argument of the Select method.

  Note: When you edit a test during a recording session, UFT does not retrieve the possible argument values from the application.

- If you assign an object to a variable, and then type the name of the variable followed by a period, UFT displays a list of the operations and properties available for the object.

  In some cases, the value of a variable cannot be determined while editing the test (for example, if the value is set by a conditional assignment or returned by another function). In this case, UFT provides statement completion information according to the most recent line of code in which the value of the variable could be evaluated, if any.

  The following examples illustrate this:

  Example 1:

  Line 1: Set x = CreateObject("Excel.Application")
  Line 2: z = GetValueFromUser()
  Line 3: If z = 2 Then
  Line 4:   Set x = CreateObject("Word.Application")
  Line 5: End If
  Line 6: x.

  While editing this test, UFT cannot determine which object will actually be assigned to x in line 6. However, because the value of x can be evaluated independently in line 4, UFT displays the statement completion information relevant to the object "Word.Application" for the variable x in line 6.

  Example 2:

  Line 1: Set x = CreateObject("Excel.Application")
  Line 2: Set x = MyGetObject()
  Line 3: x.

  While editing this test, UFT cannot determine the type of object that the **MyGetObject** function returns (line 2). Therefore, in line 3 in the example above, UFT displays the statement completion information relevant to the object "Excel.Application", because line 1 is the most recent line of code in which the value of x could be evaluated. However, if line 2 were not preceded by a line in which the value could be evaluated, UFT would not display any statement completion information for x in line 3.

Statement Completion Example for GUI Testing

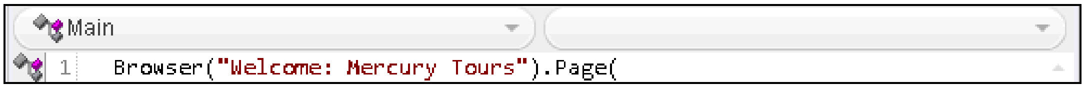
Relevant for: GUI actions, scripted GUI components, and function libraries

These examples demonstrate using the statement completion functionality step-by-step:

- "Use statement completion in the Editor" below
- "Use statement completion in a function library" on page 315

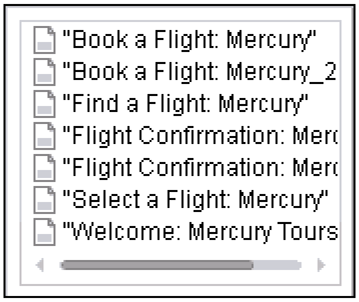
Use statement completion in the Editor

1. In the Editor, type an object followed by an open parenthesis **(**.



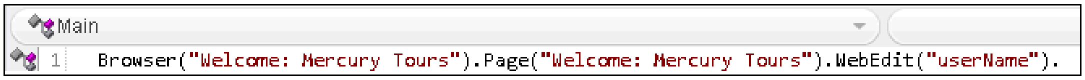
```
Browser("Welcome: Mercury Tours").Page(
```

If there is only one object of this type in the object repository, UFT automatically enters its name in quotes after the open parenthesis. If more than one object of this type exists in the object repository, UFT displays them in a list.


"Book a Flight: Mercury"
"Book a Flight: Mercury_2
"Find a Flight: Mercury"
"Flight Confirmation: Merc
"Flight Confirmation: Merc
"Select a Flight: Mercury"
"Welcome: Mercury Tours

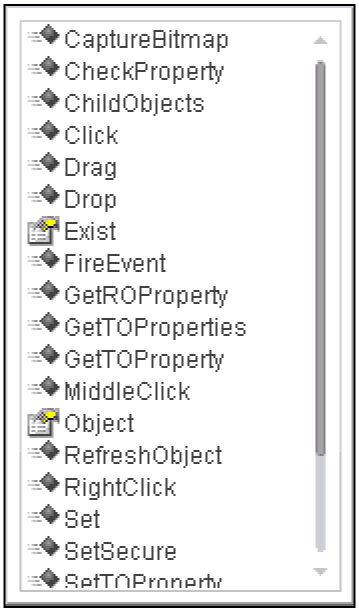
2. Double-click an object in the list or use the arrow keys to choose an object and press ENTER or SPACE. UFT inserts the object into the statement.

3. Type a period (.) after the object on which you want to perform the operation. Type the period after the object description, for example ("username").



```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").
```

UFT displays a list of the available operations and properties for the object.

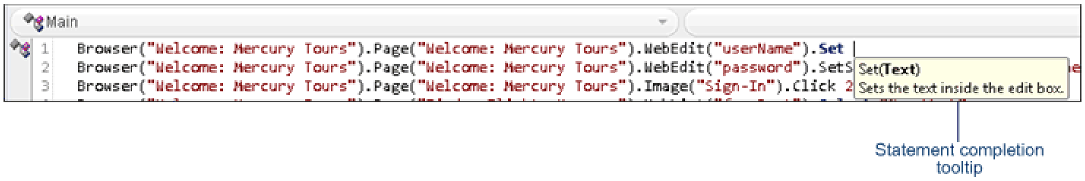


As you type the name of an operation or property, UFT highlights the first item (alphabetically) that matches the text you typed. Pressing ENTER or SPACE inserts the highlighted word into the step.

4. Double-click an operation in the list or use the arrow keys to choose an operation and press ENTER or SPACE.

   UFT inserts the operation into the statement. If the operation contains arguments, UFT displays the syntax of the operation in a tooltip.

   In the following example, the **Set** method has one argument, called **Text**. The argument name represents the text to insert in the box.



```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set |
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetS
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click 2
```

Main
Set(Text)
Sets the text inside the edit box.
Statement completion tooltip

5. Enter the operation arguments after the operation according to the displayed syntax. For example:

```
Browser("Welcome:Mercury Tours").Page("Book a Flight:Mercury").WebEdit("username").Se
t."mercury"
```

After you add a step in the Editor, you can view the new step in the Keyword View. If the statement that you added in the Editor contains syntax errors, UFT displays the errors in the

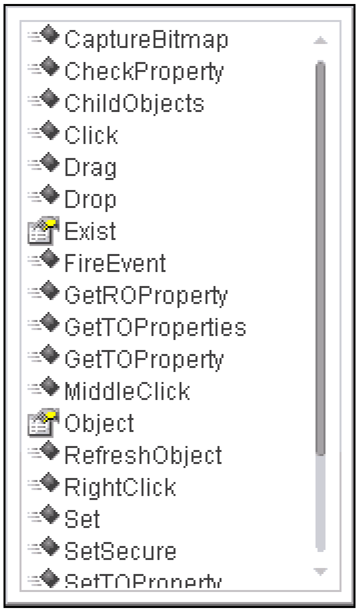
Errors pane when you select the Keyword View. For details, see "Errors Pane User Interface" on page 374.

Use statement completion in a function library

1. Type the full hierarchy of a test object, for example:

```
Browser("Welcome: Mercury Tours").Page("Book a Flight:Mercury).WebEdit("username")
```

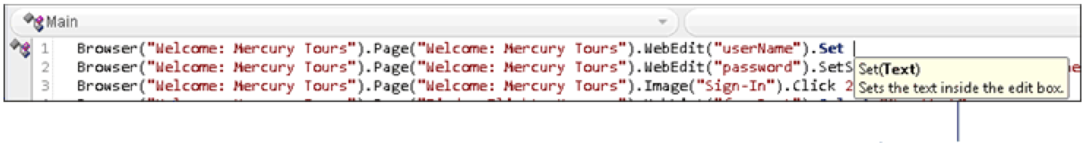
2. Type a period (.) after the object description, for example ("username"). UFT displays a list of the available operations and properties for the object.



CaptureBitmap
CheckProperty
ChildObjects
Click
Drag
Drop
Exist
FireEvent
GetROProperty
GetTOProperties
GetTOProperty
MiddleClick
Object
RefreshObject
RightClick
Set
SetSecure
SetTOProperty

3. Double-click an operation in the list or use the arrow keys to choose an operation and press ENTER or SPACE. UFT inserts the operation into the statement. If the operation contains arguments, UFT displays the syntax of the operation in a tooltip.

In the following example, the **Set** method has one argument, called **Text**. The argument name represents the text to insert in the box.

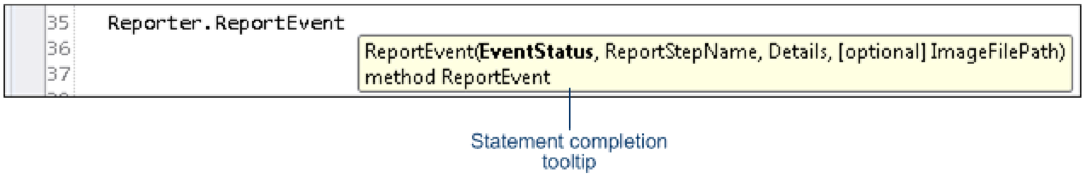


```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetS
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click 2
```

Set(**Text**)
Sets the text inside the edit box.

Statement completion tooltip

In the following example, the **ReportEvent** method has four arguments.



```
Reporter.ReportEvent
```

ReportEvent(**EventStatus**, ReportStepName, Details, [optional] ImageFilePath)
method ReportEvent

4. Enter the operation arguments after the operation.

Automatic Code Completion

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

UFT provides automatic code completion features, to facilitate coding in the Editor. This includes some basic, static code snippets that you can insert automatically into your document, as well as templates that you can use to insert additional code snippets by typing specific keywords.

For example, if you enter the letters if at the beginning of an empty line in a GUI action, followed by a SPACE, UFT automatically enters:

```
If True Then
End If
```

The word True is highlighted, reminding you to replace it with the relevant condition.

You can also modify the templates provided, or build your own customized templates as needed, and define the keywords used to invoke the use of each template.

For example, you might repeat a complicated **If...Then** statement many times your document. You can use an existing template for the **If...Then** statement to create your own customized template with your more complicated code. You can also create templates from scratch, such as a comment block template, which might include information such as programmer identification, date added, or other details you want included in all comments.

Code templates are defined in the "Code Templates Pane (Options Dialog Box > Coding Tab)" (described on page 573), and are supported for the following file types, used for actions and function libraries:

- .txt files
- For GUI testing: .mts, .qfl, or .vbs files
- For API testing: .cs files

For details, see "How to Use Code Snippets and Templates" on page 322.

For GUI testing only: If you enter two characters that are the initial characters of multiple VBScript keywords, a drop-down menu appears with all of the relevant keywords, and you can select the one you want. For example, if you enter the letters pr and then enter a space, the drop-down menu is displayed, containing the keywords preserve, private, and property. You can then select a keyword from the list and press ENTER to insert it into your script.

Searching and Replacing in the Editor

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

When searching in GUI tests you can search in the Editor for text strings only. When searching API tests, you can search in the Editor for text strings, as well as references, derived classes, base classes, or overriding methods, for the current method, function or class.

When searching for text, you can use standard text or regular expressions in your search strings, and you can perform string replacements. You can also search in documents that are closed but accessible by the search functionality, either by searching an entire solution, or specifying a search folder.

For details, see "How to Find or Replace Strings in Files" on page 327 and "How to Search for References or Classes in Documents in the Editor (API Testing Only)" on page 325.

> **Note:** Search and replace functionality is not available in the Keyword View, the canvas, or an application area.

For details, see:

- " Regular Expressions in the Find and Replace Dialog Boxes" on page 357
- "File and Item Types Included in String Searches" below

File and Item Types Included in String Searches

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

When performing text searches or replacements you can search throughout an entire solution, test, or folder. However, the specific file and item types searched within the solution, test, or folder are defined by the search algorithm and cannot be modified by users.

> **Note:** The search is limited to the text in the file at the time that the Find or Replace dialog box was opened. Any changes made after opening the dialog box are not included the search.

Searches for strings in GUI tests and scripted components

When you search for text strings in GUI tests, you can search in actions, function libraries, *.vbs files, or *.txt files. The search is performed in the action scripts, for each action defined in the test.

When you search in scripted components, the search is performed in any defined function libraries.

Searches for strings in API tests

When you search for text strings inAPI tests, you can search in actions, *.cs files, or *.txt files. The search is performed in each source code module and in the test flow.

When you search in a specific C# source code file, the search is performed throughout all user code in theAPI test, as a single text file.

You can search for the following types of items in API tests:

- Activity or event display names or event handles
- Global environment variable display names and values
- Link expressions
- Loaded XML or schema files
- Test setting definitions
- Visible checkpoint or property display names and values
- X-paths

Regular Expressions Overview

Relevant for: GUI tests and components and API testing

A **regular expression** is a string that specifies a complex search phrase. By using special characters, such as a period (**.**), asterisk (*****), caret (**^**), and brackets (**[ ]**), you can define the conditions of a search.

Regular expressions are used to identify objects and text strings with varying values. You can use regular expressions to instruct UFT to find a value that matches a particular pattern or condition instead of a specific hard-coded value.

Use case examples:

- Suppose the name of a window's title bar changes according to a file name. You can use a regular expression in a test object description to identify a window whose title bar has the specified product name, followed by a hyphen, and then any other text. When the relevant step runs, UFT compares the regular expression that you provide with the corresponding value in your application.
- Suppose the text property of an object is a date value, but the displayed date changes according to the current date. You can define a regular expression for the date, so that UFT can identify the object that contains text with the expected date format, rather than the exact date value.

Whenever a UFT feature supports regular expressions, the relevant dialog box includes a **Regular Expression** check box. Selecting this check box instructs UFT to treat the provided value as a regular expression. Some dialog boxes that contain a **Regular Expression** check box, also contain a right arrow adjacent to the text box for the value. Clicking this arrow enables you to select regular expression characters from a drop-down list, and to test your regular expression to make sure it suits your needs. For more details, see "Smart Regular Expression List" on page 361 and "Regular Expression Evaluator" on page 358.

The following are some examples of situations in which you can use regular expressions in GUI testing:

- When defining the property values of an object in dialog boxes or in programmatic descriptions used within a function library
- When defining expected values for checkpoints in tests
- When defining pop-up window conditions in a recovery scenario

You can use regular expressions only for values of type **string**.

For details on defining regular expressions, including regular expression syntax, see "Regular Expression Characters and Usage Options" on page 352.

For GUI testing only: This section also includes:

Regular Expressions for Property Values

Relevant for: GUI tests and components

If you expect the value of an object property in your application to change in a predictable way during each run session, you can use regular expressions when defining identification property values, for example, in the Object Repository window, or in programmatic descriptions. For details on programmatic descriptions, see "Programmatic Descriptions" on page 995.

For example, your Web site may include a form in which the user inputs data and clicks the **Send** button to submit the form. When a required field is not completed, the form is displayed again for the user to complete. When resubmitting the form, the user clicks the **Resend** button. You can define the value of the button's **name** property as a regular expression, so that this variation in the button name is ignored when identifying the button in your application.

> **Tip:** For tests and scripted components, UFT provides a tool to test your regular expressions to help you make sure that they suit your needs. For more details, see "Regular Expression Evaluator" on page 358. UFT provides a list of commonly used regular expression characters that you can select when creating a regular expression. For more details, see "Smart Regular Expression List" on page 361.

Regular Expressions in Checkpoints

Relevant for: GUI tests and components

When creating a standard checkpoint to verify the property values of an object, you can set the expected value of an object's property as a regular expression so that an object with a varying value can be verified.

For example, suppose you want to check that every window and dialog box in your application contains the name of your application followed by a hyphen (-) and a descriptive title. You can add a checkpoint for each dialog box object in your test to check that the first part of the title contains the name of your application followed by a hyphen.

Additional Information for GUI Tests

When creating a text checkpoint to check that a varying text string is displayed on your application, you can define the text string as a regular expression.

For example, when booking a flight in the Mercury Tours sample Web site, the total cost charged to a credit card number should not be less than $300. You define the amount as a regular expression, so that variations in the text string will be ignored as long as the value is not less than $300.

You can apply the same principles to any checkpoint type whose dialog box contains a **Configure Value** area similar to that described in "Configure Value Area " on page 1575.

For example, for table checkpoints you can set cell values as regular expressions, and for XML checkpoints you can set attribute or element values as regular expressions. For details on specific checkpoint types, see the relevant chapter for the checkpoint type.

> **Tip:** For tests and scripted components, UFT provides a tool to test your regular expressions to help you make sure that they suit your needs. For more details, see "Regular Expression Evaluator" on page 358. UFT provides a list of commonly used regular expression characters that you can select when creating a regular expression. For more details, see "Smart Regular Expression List" on page 361.

Tasks

For details about how to modify business process test and flows, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

How to Use Bookmarks in the Editor

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This task describes how to use bookmarks in testing documents and includes the following steps:

- "Insert bookmarks into a document" below
- "Navigate between bookmarks and to specific bookmarks" below
- "Clear bookmarks" below

Insert bookmarks into a document

1. Create or open an action, scripted GUI component, function library, or user code file. For details, see "How to Create and Manage Documents" on page 136.

   (If you are editing a GUI action or scripted GUI component, open it in the Editor.)

2. Click in the line to which you want to assign a bookmark.

3. Select **Search > Bookmarks > Toggle Bookmarks**, or press the Bookmarks button in the Bookmarks pane. A bookmark icon □ is added to the left of the selected line and is also displayed in the "Bookmarks Pane" (described on page 203).

Navigate between bookmarks and to specific bookmarks

1. To navigate between bookmarks listed in the "Bookmarks Pane", do one of the following:

   - Select **Search > Bookmarks > Next Bookmark** or **Search > Bookmarks > Previous Bookmark** to navigate forward and backward, respectively.
   - In the "Bookmarks Pane", click the **Next Bookmark** and **Previous Bookmark** ⇦ ⇨ buttons.

2. To navigate to a specific bookmark, double-click the bookmark's line in the "Bookmarks Pane".

Clear bookmarks

Do one of the following:

- Click a bookmark □ icon to the left of the selected line to delete the bookmark.
- In the "Bookmarks Pane", click the **Delete**

button to delete the selected bookmark.

- In the "Bookmarks Pane", select **Search > Bookmarks > Clear All Bookmarks** or click the **Delete All** button in the "Bookmarks Pane" to delete all bookmarks.

How to Use the Go To Dialog Box in the Editor

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This task describes how to go to a specific line of code, API test class, or function, in a GUI action, scripted GUI component, function library, or user code, or to navigate to any file.

1. Create or open anaction, scripted GUI component, function library, oruser code file. For details, see "How to Create and Manage Documents" on page 136.

   (If you are editing a GUI action or scripted GUI component, open it in the Editor.)

2. In the Editor, select **Search > Go To > Location**. The "Go To Dialog Box" (described on page 347) opens.

> **Tip:** By default, line numbers are displayed in the Editor. If they are not displayed, you can select the **Show line numbers** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node). For details on the Text Editor options, see "General Pane (Options Dialog Box > Text Editor Tab)" on page 575.

How to Use Code Snippets and Templates

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This task describes how to insert pre-designed code snippets or blocks of text into your document, as well as how to manage templates for such snippets in the "Code Templates Pane (Options Dialog Box > Coding Tab)" (described on page 573). For details on code snippets and templates, see "Automatic Code Completion" on page 316.

This task includes the following steps:

- "Insert code snippets into your document in the Editor" below
- "Modify an existing list of code templates" on the next page
- "Remove an existing list of code templates" on page 324
- "Add a new list of code templates" on page 324

Insert code snippets into your document in the Editor

1. Create or open an action, scripted GUI component, function library, or user code file. For

details, see "How to Create and Manage Documents" on page 136.

(If you are editing a GUI action or scripted GUI component, open it in the Editor.)

2. **In an API test:** Place your cursor at the point in the file that you want to insert the code snippet, type the keyword for a code template defined in the Code Templates pane of the Options dialog box, and press TAB.

   In a GUI test or scripted component: Place your cursor at the point in the document that you want to insert the code snippet and then enter text as described in the following table.

If you enter:	Followed by:	UFT does the following:
The keyword for a code snippet listed in the **Edit > Code Snippet** menu (GUI tests and scripted components)	**SPACE**	Inserts the first VBScript snippet defined for the keyword you entered, as listed in the **Edit > Code Snippet** menu. **Note:** You can also insert these snippets by selecting them from the **Edit > Code Snippet** menu.
The keyword for a code template defined in the Options dialog box	**TAB**	Inserts the code snippet defined in the relevant template in the "Code Templates Pane (Options Dialog Box > Coding Tab)" (described on page 573). **Note:** If multiple templates are defined for the keyword you entered, the first template in the list is inserted.
Part of a code template keyword	**CTRL+SPACE**	Displays a list of the code snippets and templates whose keywords match the characters you entered. The list includes the static VBScript snippets listed in the **Edit > Code Snippet** menu, as well as the code templates defined for the relevant file type in the "Code Templates Pane (Options Dialog Box > Coding Tab)"(described on page 573). In the drop-down list, browse to the keyword for the template you want to insert, and press Tab to insert its snippet into your document.

Modify an existing list of code templates

1. Open the "Code Templates Pane (Options Dialog Box > Coding Tab)" (described on page 573).

2. From the **File Types** drop-down list, select the item associated with the list of templates you want to modify. The table lists all code templates defined for the selected file type or types.

3. To edit the file types associated with the selected list, click **Edit List**. In the Edit List dialog box, enter the file type or types that should be supported by the selected list, separated by semi-colons (;).

4. To edit the list of code templates, select the table row for the code template you want to modify and do one of the following:

   - **Add a new template in the list:** Click the empty space below the last row in the table, above the syntax area.

   - **Edit the template name:** Double-click the cell in the Template column, and update the name.

   - **Edit the keyword:** Double-click the cell in the Keyword column, and update the keyword. The keyword is the text that you enter in the Editor to insert the template.

   - **Edit the code template description:** Double-click the cell in the Description column and update the description text.

   - **Edit the code syntax inserted into your code:** Click inside the code syntax area below the table and update the code.

   > **Note:** Note that changes made here do not affect the code snippets available from the **Edit > Code Snippet** menu, which are static and cannot be modified.

Remove an existing list of code templates

1. Open the "Code Templates Pane (Options Dialog Box > Coding Tab)", described on page 573.

2. From the **File Types** drop-down list, select the item associated with the list of templates you want to remove. The table lists all code templates defined for the selected file type or types.

3. Click **Remove List**. All code templates associated with the selected file types are removed, as well as the **File Types** item.

   > **Caution:** This action is irreversible.

Add a new list of code templates

1. Open the "Code Templates Pane (Options Dialog Box > Coding Tab)", described on page 573.

2. Click **Add List**.

3. In the Add List dialog box, enter the file types you want to associate with your new list of templates, separated by semi-colons (;). A blank list is added to the table.

4. In each of the cells in the rows, enter text to add template names, keywords to be entered in the code, and descriptions of each template. In the syntax area below the table, enter the code template to be inserted in your code.

5. To add a new row in the list, click the empty space below the last row in the table, above the syntax area.

How to Search for References or Classes in Documents in the Editor (API Testing Only)

Relevant for: User code files

This task describes how to search for references to functions or method definitions, or base or descending classes, and includes the following steps:

- "Search for references to the currently selected function or method" below
- "Search for classes derived from the currently selected class" below
- "Search for methods that override a virtual method" on the next page
- "Search for the base class of the current class" on the next page

Search for references to the currently selected function or method

1. Create a new user code file, or open an existing one. For details, see "Add an event handler - optional" on page 1601 or "How to Open a Window for Writing Custom Code" on page 1948.

2. Select a function or method definition, and then select **Search > Find References**.

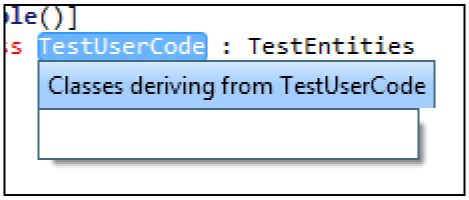
   The search results found are displayed in the "Search Results Pane" (described on page 466).

Search for classes derived from the currently selected class

1. Create a new user code file, or open an existing one. For details, see "Add an event handler - optional" on page 1601 or "How to Open a Window for Writing Custom Code" on page 1948.

2. Select a class and then select **Search > Find Derived Symbols**.

The derived classes are displayed in a small drop-down box under the selected class. For example:

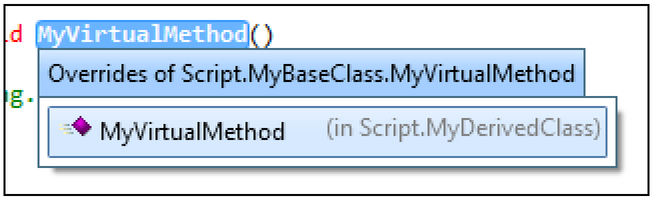

le()]
s TestUserCode : TestEntities
Classes deriving from TestUserCode

Search for methods that override a virtual method

1. Create a new user code file, or open an existing one. For details, see "Add an event handler - optional" on page 1601 or "How to Open a Window for Writing Custom Code" on page 1948.

2. Select a class and then select **Search > Find Derived Symbols**.

   The overriding methods are displayed in a small drop-down box under the selected class. For example:

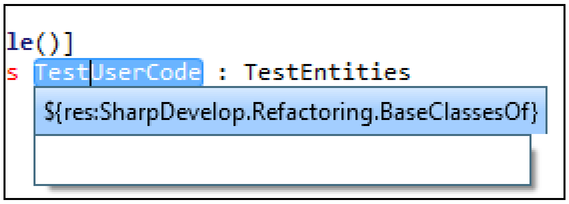

d MyVirtualMethod()
g.
Overrides of Script.MyBaseClass.MyVirtualMethod
MyVirtualMethod (in Script.MyDerivedClass)

Search for the base class of the current class

1. Create a new action or user code file, or open an existing one. For details, see "Add an event handler - optional" on page 1601, "How to Open a Window for Writing Custom Code" on page 1948, or "How to Use Actions in an API Test" on page 1793.

2. Select a class and then select **Search > Find Base Classes**.

   The base classes are displayed in a small drop-down box under the selected class. For example:


le()]
s TestUserCode : TestEntities
${res:SharpDevelop.Refactoring.BaseClassesOf}

How to Find or Replace Strings in Files

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This task describes how to search for strings in files that are open in the document pane or supported by the search functionality and includes the following steps:

- "Define the search or replace criteria" below
- "Search for strings in files" on the next page
- "Replace strings in files" on page 329

You can find individual occurrences, display all search results, or replace occurrences of the search string with a different string. If a search generates multiple results, all results are displayed in the "Search Results Pane" (described on page 466).

> **Note:** You cannot modify the file and item types included in the search, which are defined by the search algorithm. For details, see "File and Item Types Included in String Searches" on page 317.

Define the search or replace criteria

1. Create or open an action, scripted GUI component, function library, or user code file. For details, see "How to Create and Manage Documents" on page 136.

   (If you are editing a GUI action or scripted GUI component, open it in the Editor.)

2. Do one of the following:

   - To only find strings, open the "Find Dialog Box (Document Pane Editor) ", described on page 342.
   - To find and replace strings, open the "Replace Dialog Box", described on page 344.

3. In the **Find text** field, enter a search string using plain text or regular expressions. For details, see " Regular Expressions in the Find and Replace Dialog Boxes" on page 357.

4. If you entered a regular expression manually, select the **Regular Expression** check box. (If you selected a regular expression from the **Expression Builder** drop-down list, the **Regular Expressions** check box is automatically selected.)

5. **For replacements only:** In the **Replace with** field, enter the replacement string.

6. From the **Look in** drop-down list, select a location in which to search, or search and replace. Supported locations include the current solution, test, document, or any specified folder, except for folders stored in ALM. If you want to search inside a document stored in ALM, you must open the document first.

7. Select any of the other search options as needed, including matching the text case, matching the whole word, searching upwards in the file instead of downwards, or including subfolders if you selected a folder to search.

Search for strings in files

To search for occurrences of the string one by one:

Do one of the following:

- In the "Find Dialog Box (Document Pane Editor) " (described on page 342), click **Find Next.**
- To search again for the most recently defined search criteria without opening the Find dialog box, select **Search > Find Next** or press F3.

The cursor jumps to the next occurrence of the search string and highlights the string. If the search string is found in a closed file, the file automatically opens in the document pane.

> **Note: (for GUI tests only)** If you switch to the Keyword View, the entire row containing the matched string is highlighted.

To search for all occurrences of the search string:

In the Find dialog box, click **Find All.** The search results are displayed in the "Search Results Pane" (described on page 466).

To perform an incremental search as you type:

1. Do one of the following:
   - To search from the cursor location towards the end of the file, select **Search > Incremental Search** or press CTRL+E.
   - To search from the cursor location towards the top of the file, select **Search > Reverse Incremental Search** or press CTRL+SHIFT+E.

   The cursor changes to a binoculars icon with an arrow pointing in the search direction.
2. Start typing the string you want to find. UFT highlights the next matching string.
3. Click anywhere in the file to change the cursor back to the regular cursor.

   > **Note:** When performing incremental searches, you can find only one search result at a time. Repeat this step to find the next occurrence of the search string.

Replace strings in files

To replace occurrences of the string one by one:

1. In the "Replace Dialog Box" (described on page 344), click **Find Next** until you reach the search result you want to replace.

2. Click **Replace**. The selected string is replaced with the defined replacement text. The next occurrence is automatically highlighted.

To replace all occurrences of the search string:

In the Replace dialog box, click **Replace All**. A confirmation message lists the number of occurrences replaced.

Reference

Document Pane User Interface

Relevant for: GUI tests and components and API testing

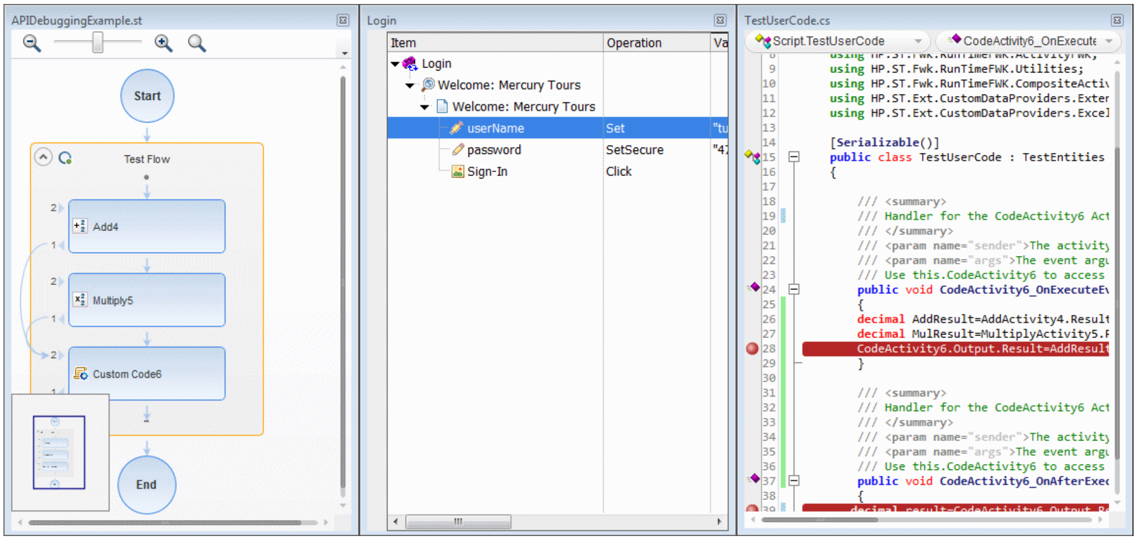
This pane enables you to view and edit UFT documents. For details on the types of available documents, see "Document Pane Overview" on page 303.

The image below shows the various document views available in UFT, including (from left to right):

- An API test (canvas)
- A GUI action (Keyword View)
- The event handler code file (TestUserCode.cs) (Editor)

All UFT documents are shown in the document pane. For other examples, see "Application Area User Interface" on page 2101 and "Business Process Testing in UFT User Interface" on page 2074.

> **Note:** The documents shown in the image below are undocked from the document pane. By default, these views are displayed as tabs in the document pane.



Item	Operation	Va
Login		
Welcome: Mercury Tours		
Welcome: Mercury Tours		
userName	Set	"tu
password	SetSecure	"47
Sign-In	Click	

```
    using HP.ST.Fwk.RunTimeFWK.ActivityFWK;
    using HP.ST.Fwk.RunTimeFWK.Utilities;
    using HP.ST.Fwk.RunTimeFWK.CompositeActi
    using HP.ST.Ext.CustomDataProviders.Exter
    using HP.ST.Ext.CustomDataProviders.Excel

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Act
        /// </summary>
        /// <param name="sender">The activity
        /// <param name="args">The event argu
        /// Use this.CodeActivity6 to access
        public void CodeActivity6_OnExecuteEv
        {
        decimal AddResult=AddActivity4.Result
        decimal MulResult=MultiplyActivity5.F
        CodeActivity6.Output.Result=AddResult
        }

        /// <summary>
        /// Handler for the CodeActivity6 Act
        /// </summary>
        /// <param name="sender">The activity
        /// <param name="args">The event argu
        /// Use this.CodeActivity6 to access
        public void CodeActivity6_OnAfterExec
        {
        decimal result=CodeActivity6.Output.Re
```

To access	Do one of the following: • Create or open a document. • Double-click a node in the Solution Explorer. A document opens as a tab in the document pane.

Important information	• If you select a document in the Solution Explorer, but do not double-click it to bring it into focus in the document pane, some other panes (such as the Properties pane), automatically display content relevant for that document. Therefore, the document displayed in the document pane may not correspond to the content of other open panes. • When you bring a document into focus, the corresponding node in the Solution Explorer is highlighted, and other panes are updated accordingly.
Relevant tasks	"How to Create and Manage Documents" on page 136
See also	• "Document Pane Overview" on page 303 • "Document Pane Context Menu Options" on the next page • "Editor User Interface" on page 334

User interface elements for the document when documents are displayed as tabs are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<tab name>	The name of the document. If you hover over the name of the tab, you can view information about the selected document: • **For actions:** the test in which the action is located • **For function libraries and user code files**: the path on the file system in which the document is saved • **For application areas:** the ALM path of the application area
⏷	Displays a drop-down list of all documents currently open in the document pane.
☒	Closes the active document.

User interface elements for the document when documents are displayed as floating documents are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<window name>	The name of the document.
☒	Closes the active document.

Document Pane Context Menu Options

Relevant for: GUI tests and components and API testing

This section describes the context menu options available for documents displayed as tabs or document windows in the documents pane.

To access	Right-click the document tab or title bar (for floating documents).
Important information	This information describes the context menu options for general document pane tabs and floating documents. For details on the context menu options for a specific type of document, see the relevant section in this guide.

The context menu options for documents displayed as tabs are described below:

Menu Command	Keyboard Shortcut	Description
Close	CTRL+F4	Closes the current document.
Close All Documents	CTRL+SHIFT+ F4	Closes all documents.
Close All but This		Closes all documents except the current document.
Save	CTRL+S	Saves the current document. **Note:** This command is enabled only if you have modified a document.
Save As		Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164) so you can save a copy of the current document. with another name or in another location. **Note:** This option is enabled only if a document is able to be saved under a different name. For example, you can perform **Save as** for a test but not a test action. This option is only available if the open document is an editable file (such as aGUI test action, user code file, or function library.)

Menu Command	Keyboard Shortcut	Description
Copy file/path name		Copies the full path of the document in the file system or an ALM project to the Clipboard. **Note:** This option is only available if the open document is an editable file (such as aGUI test action, user code file, or function library.)
Open Containing Folder in Explorer		Opens the folder containing the document in Windows Explorer. **Note:** This option is only available if the open document is an editable file (such as aGUI test action, user code file, or function library.)

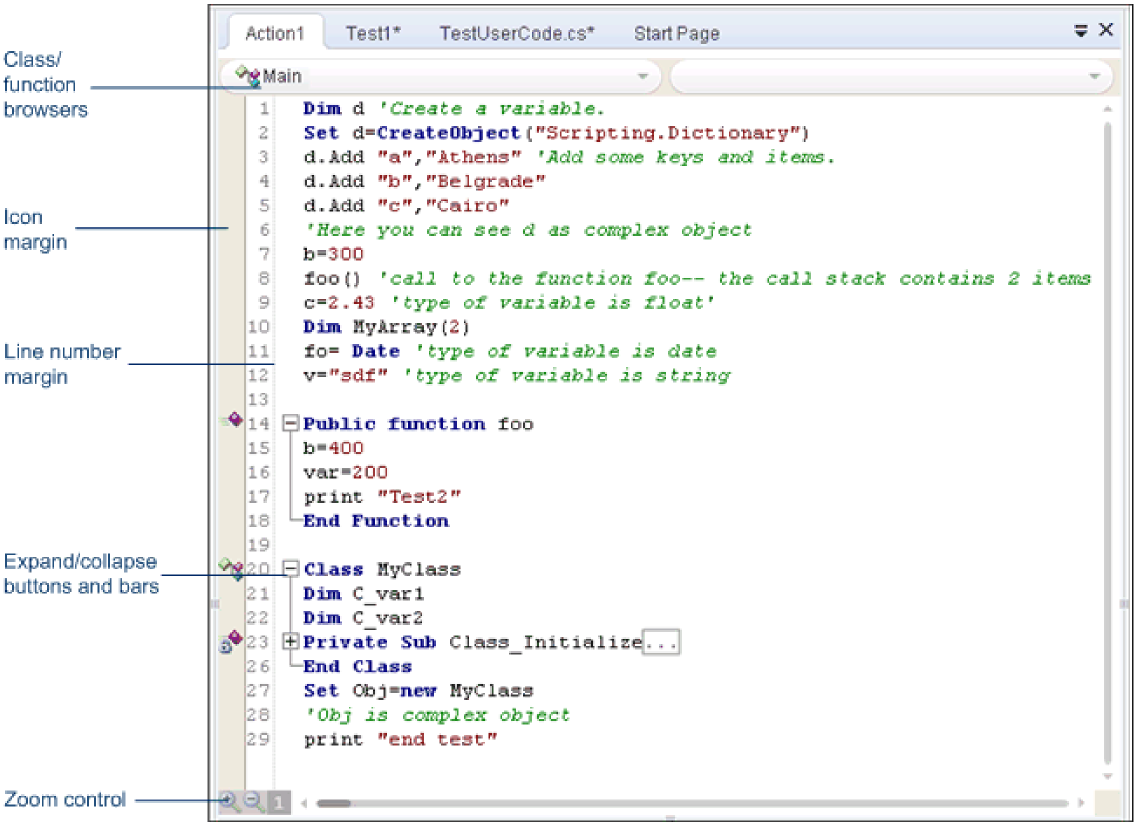
The context menu options for documents displayed as floating windows are described below:

UI Element	Description
Close	Closes the document.
Dock as Tabbed Document	Attaches the document as a tab in the document pane.

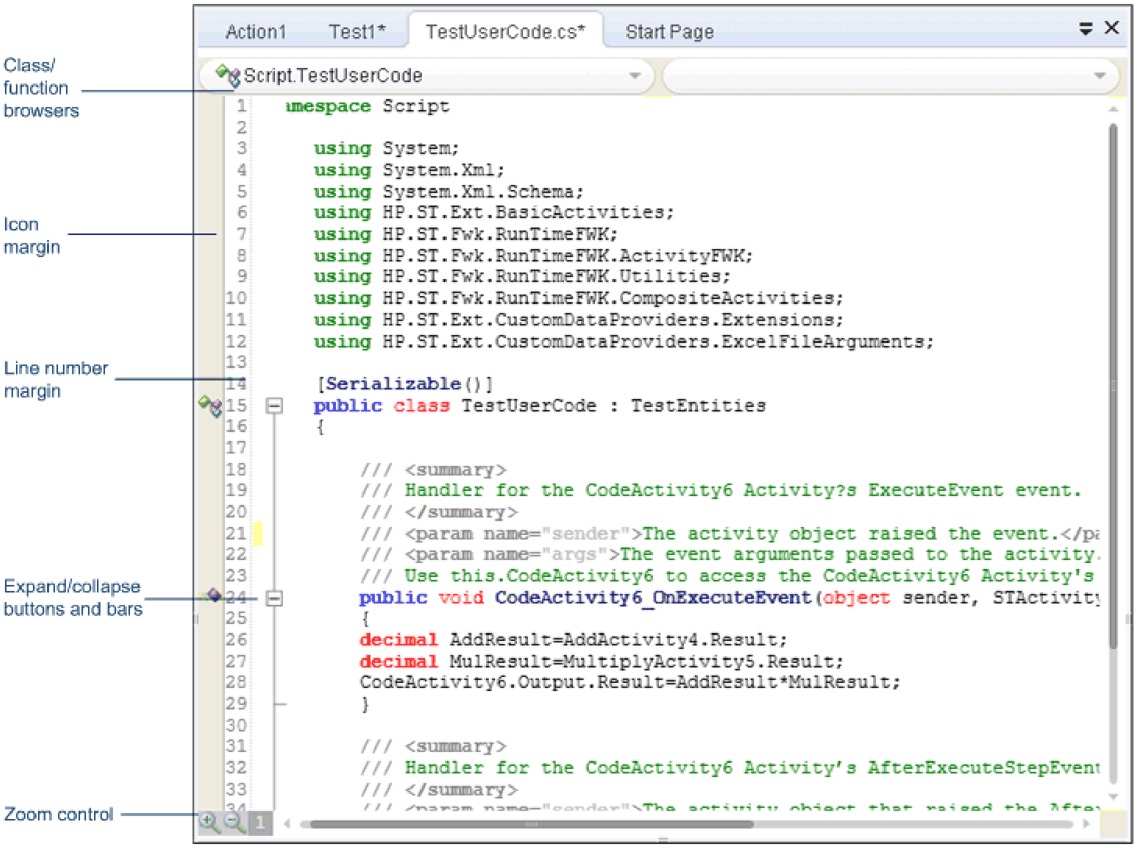
Editor User Interface

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This view enables you to edit text and code for GUI actions, scripted GUI components, function libraries, user code files, and text files.



```
Dim d 'Create a variable.
Set d=CreateObject("Scripting.Dictionary")
d.Add "a","Athens" 'Add some keys and items.
d.Add "b","Belgrade"
d.Add "c","Cairo"
'Here you can see d as complex object
b=300
foo() 'call to the function foo-- the call stack contains 2 items
c=2.43 'type of variable is float'
Dim MyArray(2)
fo= Date 'type of variable is date
v="sdf" 'type of variable is string

Public function foo
b=400
var=200
print "Test2"
End Function

Class MyClass
Dim C_var1
Dim C_var2
Private Sub Class_Initialize...
End Class
Set Obj=new MyClass
'Obj is complex object
print "end test"
```



```
umespace Script

    using System;
    using System.Xml;
    using System.Xml.Schema;
    using HP.ST.Ext.BasicActivities;
    using HP.ST.Fwk.RunTimeFWK;
    using HP.ST.Fwk.RunTimeFWK.ActivityFWK;
    using HP.ST.Fwk.RunTimeFWK.Utilities;
    using HP.ST.Fwk.RunTimeFWK.CompositeActivities;
    using HP.ST.Ext.CustomDataProviders.Extensions;
    using HP.ST.Ext.CustomDataProviders.ExcelFileArguments;

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s ExecuteEvent event.
        /// </summary>
        /// <param name="sender">The activity object raised the event.</pa
        /// <param name="args">The event arguments passed to the activity.
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's
        public void CodeActivity6_OnExecuteEvent(object sender, STActivity
        {
        decimal AddResult=AddActivity4.Result;
        decimal MulResult=MultiplyActivity5.Result;
        CodeActivity6.Output.Result=AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity's AfterExecuteStepEvent
        /// </summary>
        /// <param name="sender">The activity object that raised the After
```

To access	Create a new action, scripted GUI component, function library, or user code file, or open an existing one. (If you are editing a GUI action or scripted GUI component, open it in the Editor.)
Important information	To open the help topic for a specific object or method from the Editor, do the following: 1. Verify that you do not have any text highlighted. 2. Place the cursor in the middle of the word and press **F1**. **Note:** You can also use these steps to find VBScript items referenced in the Microsoft VBScript help, included in the UFT Help. If you place the cursor in an item that is not part of the UFT test object model and press **F1**, the UFT Help displays the item as found in the index, or if the exact item is not found, the closest alphabetical item in the index.

Relevant tasks	"How to Create and Manage Documents" on page 136 API tests only: "How to Create an API Test" on page 1598 "How to Use Bookmarks in the Editor" on page 321 "How to Use the Go To Dialog Box in the Editor" on page 322 "How to Use Code Snippets and Templates" on page 322 "How to Search for References or Classes in Documents in the Editor (API Testing Only)" on page 325 "How to Find or Replace Strings in Files" on page 327
See also	• "Editing Text and Code Documents" on page 305 • API tests only: "Writing Event Handlers for API Test Steps" on page 1935 • GUI tests only: "Programming in GUI Testing Documents in the Editor" on page 994 • "Text Editor Tab (Options Dialog Box)" on page 574 • "Find Dialog Box (Document Pane Editor) " on page 342 • "Replace Dialog Box" on page 344 • "Go To Dialog Box" on page 347 • "Bookmarks Pane User Interface" on page 206

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

UI Element	Description
<class and function browsers>	Drop-down menus that enable you to quickly browse to a class and function by selecting from those available in the current document. These drop-down menus are most helpful in documents that include many function definitions, such as function libraries. To navigate to a class or function definition, first select the class from the left drop-down menu, and then select the function from the right drop-down menu. **For GUI actions only:** In a GUI action, the class browser: • Displays only those functions actually defined in the action, and does not display all functions defined in the function libraries associated with the test. • Always displays **Main** as the root menu item. In a document that includes function definitions, the functions will be listed as sub-items in the menu. In a document that includes no function definitions (such as most actions), this menu will include only the **Main** menu item. For example, see "Class and Function Browser Examples" on the next page. **Note:** If the class and function browsers are not displayed, you can select the **Show class/function browser** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node).
<Icon margin>	Display icons for lines with specific types of code, such as classes, events, or fields. For details, see "Editor Icons" on page 339.
<Line number margin>	Displays the line numbers for each line of code. You can use the "Go To Dialog Box" (described on page 347) to navigate to a specific line in the code. **Note:** If the line numbers are not displayed, you can select the **Show line numbers** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node.
⊞ ⊟	**<Expand and collapse>.** Expands or collapses folded code. Folded code is indicated by an ellipsis at the end of the first line of the folded code. **Tip:** Hover over the ellipses to display a snapshot of the hidden code. **Note:** If the expand and collapse buttons are not displayed, you can select the **Enable folding** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node.

UI Element	Description
⊕ ⊖ 1	**<Zoom control bar>.** Displayed in the lower left corner of the Editor, only after having zoomed in or out of the text for the first time in a session. • Click the **Zoom in** ⊕ icon to zoom in. • Click the **Zoom out** ⊖ to zoom out. • Click the **Default zoom** 1 icon to return the text to the default size.

Class and Function Browser Examples

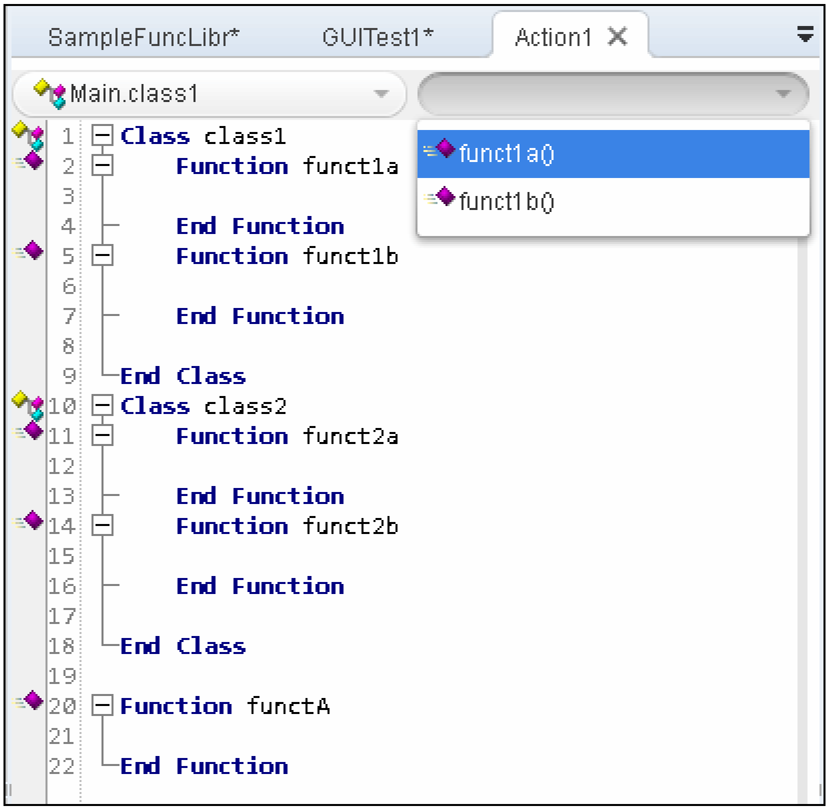
Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

The following image shows an example of an action containing two classes and a function. Each of the classes contains two functions.

The functions defined in a specific class are displayed in the function browser only when you select that class in the class browser. If you select one of these functions, the cursor jumps to that function definition. If you select a different class from the class browser, the cursor jumps to that class definition.

In this image, the **Main.class1** class is selected in the class browser, and the function browser is expanded, displaying the functions defined in the **class1** class (**fucnt_1a** and **fucnt_1b**).

In the class browser, the **Main** item represents the context of the document. In this image, if you select **Main** from the class browser, the function browser displays only **funct_A**.



```
Class class1
    Function funct1a

    End Function
    Function funct1b

    End Function

End Class
Class class2
    Function funct2a

    End Function
    Function funct2b

    End Function

End Class

Function functA

End Function
```

Editor Icons

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

Icons used in the Editor to indicate the different types of code are described below:

Type	Icon	Description
Class		Class
		Internal class
		Private class
		Protected class

Type	Icon	Description
Delegate		Delegate
		Internal delegate
		Private delegate
		Protected delegate
Enumeration definition		Enumeration definition
		Internal enumeration definition
		Private enumeration definition
		Protected enumeration definition
Event		Event
		Internal event
		Private event
		Protected event
Extension method		Extension method
		Internal extension method
		Private extension method
		Protected extension method
Field		Field
		Internal field
		Private field
		Protected field
Indexer		Indexer
		Internal indexer
		Private indexer
		Protected indexer

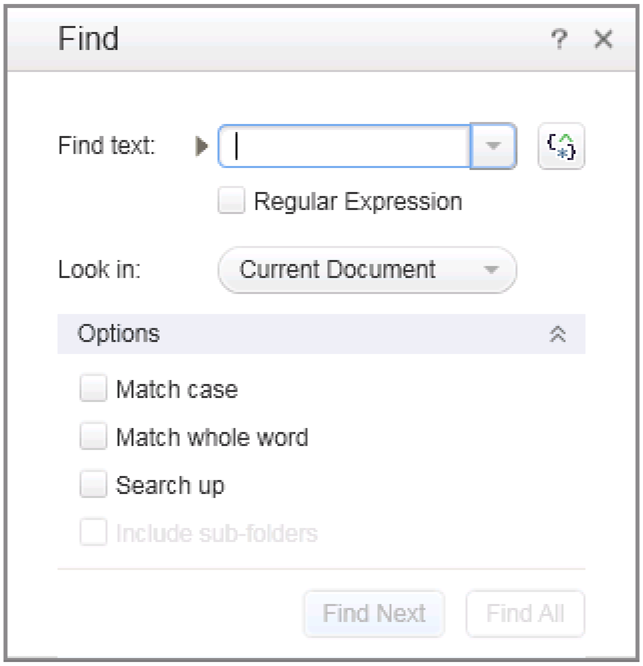
Type	Icon	Description
Interface		Interface
		Internal interface
		Private interface
		Protected interface
Keyword		Keyword
Literal		Literal
Local		Local
Method		Method
		Internal method
		Private method
		Protected method
Namespace	{ }	Namespace
Operator	<>	Operator
Parameter		Parameter
Property		Property
		Internal property
		Private property
		Protected property
Reference		Reference
Structure		Structure
		Internal structure
		Private structure
		Protected structure

Find Dialog Box (Document Pane Editor)

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This dialog box enables you to search for strings in files displayed in the Editor or located in a specific folder.

You can either find literal text or use regular expressions. You can also use other options to further fine-tune your search.



To access	1. Create or open the document in which you want to search, or, to search in multiple files, open any document. 2. Select **Search > Find** or press CTRL+F.
Important information	• You cannot use the **Find** dialog box to search in the Keyword View, the canvas, or application areas. • The search is limited to the text in the file at the time that the **Find** dialog box was opened. Any changes made after opening the **Find** dialog box are not included the search.
Relevant tasks	"How to Find or Replace Strings in Files" on page 327
See also	"Searching and Replacing in the Editor " on page 317

User interface elements are described below:

UI Elements	Description
Find text	The text string you want to locate. For details on searching with a regular expression, see the **Expression Builder** button description.
▾	**Recent searches**. Click to select a recently used search string. The list includes the last ten search strings.
{*}	**Expression Builder**. Click to select a predefined regular expression and insert it into your search string at the cursor location. When you click this button, the **Regular expression** check box is automatically selected. For details on regular expressions in search strings, see " Regular Expressions in the Find and Replace Dialog Boxes" on page 357.
Regular Expression	Treats the specified search string as a regular expression. This option is automatically selected when you select a regular expression from the Smart Regular Expression drop-down list.
Look in	The files or folder you want to search. Select one of the following from the drop-down list: • **Current Document.** Searches within the document currently open in the Editor. This is the default option. • **Current Test.** Searches within the currently open test. • **Entire Solution.** Searches within the entire solution for the file currently open in the Editor. • **<Files...>.** Enables you to browse to a select a specific folder in the file system. **Note: (for ALM users)** You cannot browse to an ALM path. The drop-down list also displays the last ten folders you selected, with the most recent folder listed first. Select one to search in the same folder again. For details, see "File and Item Types Included in String Searches" on page 317.

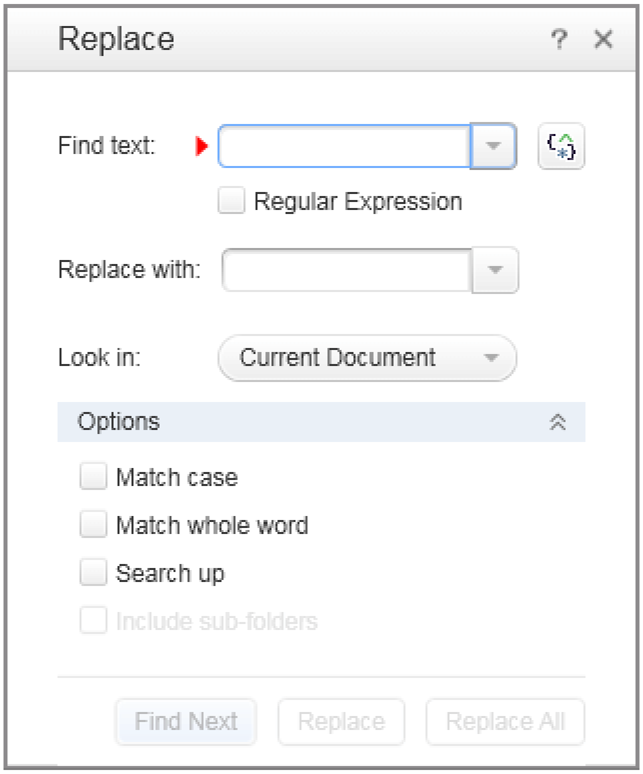
UI Elements	Description
Match case	Distinguishes between upper-case and lower-case characters in the search, and searches only for occurrences with exact matches to the casing in the search string.
Match whole word	Searches only for occurrences that are whole words and not part of longer words.
Search up	Searches from the cursor location towards the top of the file, and then wraps around to the current cursor location from the bottom of the file. By default, the search is performed downwards in the file.
Include sub-folders	Searches through any sub-folders found in the selected folder. **Note:** This option is available only when you select a specific folder from the **Look in** drop-down list.
Find Next	Searches for the next occurrence of the defined search criteria.
Find All	Searches for all occurrences of the defined search criteria. Search results are listed in the Search Results pane. For details, see "Search Results Pane User Interface" on page 466.

Replace Dialog Box

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This dialog box enables you to search for strings in files displayed in the Editor or located in a specific folder, and replace them with a different string.

You can either find and replace literal text or use regular expressions. You can also use other options to further fine-tune your find and replace process.



To access	1. Create or open the document in which you want to search, or, to search in multiple files, open any document. 2. Select **Search > Replace** or press **CTRL+H**.
Important information	• You cannot use the **Replace** dialog box to search in the Keyword View, the canvas, or application areas. • The search is limited to the text in the file at the time that the **Replace** dialog box was opened. Any changes made after opening the **Replace** dialog box are not included the search.
Relevant tasks	"How to Find or Replace Strings in Files" on page 327
See also	"Searching and Replacing in the Editor " on page 317

User interface elements are described below:

UI Elements	Description
Find text	The text string you want to locate. For details on searching with a regular expression, see the **Expression Builder** button description.

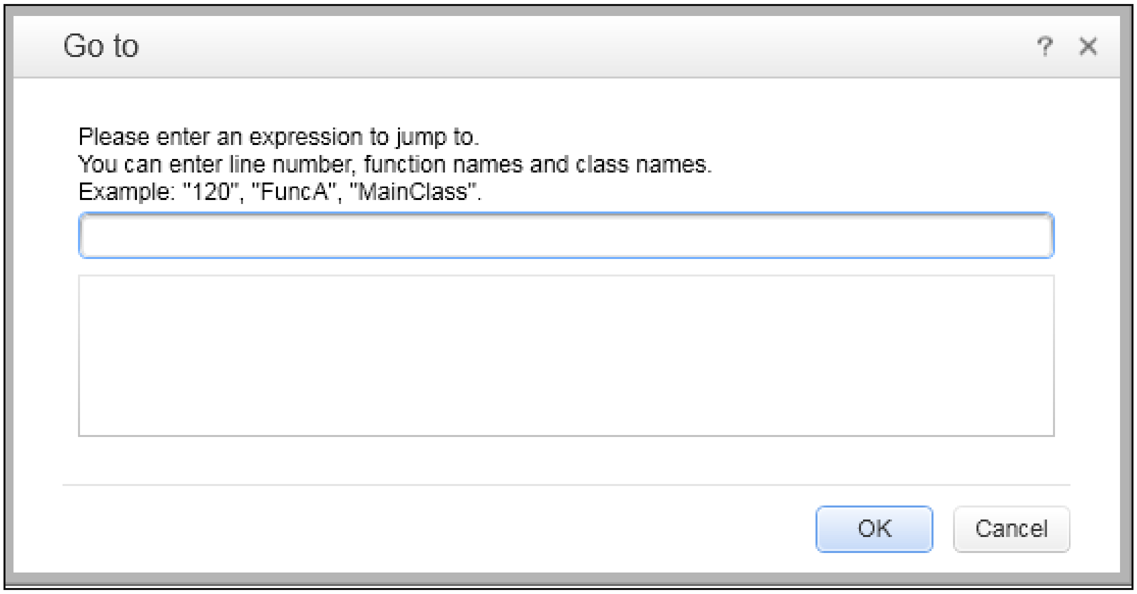
UI Elements	Description
▾	**Recent searches and replacements.** Click to select a recently used string. The list includes the last ten search strings.
{*}	**Expression Builder.** Click to select a predefined regular expression and insert it into your search string at the cursor location. When you click this button, the **Regular expression** check box is automatically selected. For details on regular expressions in search strings, see " Regular Expressions in the Find and Replace Dialog Boxes" on page 357.
Regular Expression	Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.
Replace with	The text string you want to use as the replacement string. The text is replaced using the same casing as defined in the replacement string.
Look in	The files or folder you want to search. Select one of the following from the drop-down list: • **Current Selection.** Searches within the currently highlighted text. You must highlight a specific area in the current file before selecting this option. • **Current Document.** Searches within the document currently open in the Editor. This is the default option. • **Current Test.** Searches within the currently open test. • **Entire Solution.** Searches within the entire solution for the file currently open in the Editor. • **<Files...>.** Enables you to browse to a select a specific folder in the file system. **Note: (for ALM users)** You cannot browse to an ALM path. If you want to search in a document stored in ALM, you must open the document first. The drop-down list also displays the last ten folders you selected, with the most recent folder listed first. Select one to search in the same folder again. For details, see "File and Item Types Included in String Searches" on page 317.
Match case	Distinguishes between upper-case and lower-case characters in the search, and searches only for occurrences with exact matches to the casing in the search string.

UI Elements	Description
Match whole word	Searches only for occurrences that are whole words and not part of longer words.
Search up	Searches from the cursor location towards the top of the file, and then wraps around to the current cursor location from the bottom of the file.
Include sub-folders	Searches and replaces through any sub-folders found in the selected folder. **Note:** This option is available only when you select a specific folder from the **Look in** drop-down list.
Find Next	Searches for the next occurrence of the defined search criteria.
Replace	Replaces the string defined in the **Find text** field with the string defined in the **Replace with** field.
Replace All	Replaces all occurrences of the string defined in the **Find text** field with the string defined in the **Replace with field**, throughout the location defined in the **Look in** drop-down list. All strings are found and replaced in the same action.

Go To Dialog Box

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This dialog box enables you to navigate to a specific line,API testclass, or function in aGUI action, scripted GUI component, function library, or user code file, or to navigate to any file.



Go to

Please enter an expression to jump to.
You can enter line number, function names and class names.
Example: "120", "FuncA", "MainClass".

To access	1. Create a new solution or open an existing solution. 2. Open the necessary documents. For details on how to open documents included in a solution, see "How to Manage Items in the Solution Explorer Pane" on page 473. 3. Select **Search > Go To**.
Important information	**Search Limitations:** • When you navigate to an API test class, or function name, UFT searches all files within the open solution for occurrences of the search terms. • When you navigate to line numbers, UFT searches only the currently selected document. • You can navigate to any file name contained within the current solution.
Relevant tasks	"How to Use the Go To Dialog Box in the Editor" on page 322
See also	"Bookmarks Overview" on page 204

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

UI Elements	Description
<search text>	The location to which you want to go. Enter one of the following: • A line number • An API test class name • A function name • Any file name
<search results>	The locations that match the text you entered. As you enter a navigation search string, all locations that match the current string are displayed. Double-click a specific location to navigate directly to it. If a document containing the location is closed, UFT opens the relevant document.

Editor Context Menu Options

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This section describes the context menu options available for the different document types displayed in the Editor.

To access	1. Create or open anaction, scripted GUI component, function library, oruser code file. (If you are editing a GUI action or scripted GUI component, open it in the Editor.) 2. Right-click in the document.
Important information	- Some options are also available through menu commands or toolbar buttons. - Some of the context-menu options may be hidden depending on the view of the document or the content of the document.
Relevant tasks	**For GUI testing:** "How to Enhance Your Actions, Scripted Components, and Function Libraries Using the Windows API" on page 1011 **For API testing:**"How to Open a Window for Writing Custom Code" on page 1948
See also	**For GUI testing:** - "Programming in GUI Testing Documents in the Editor - Overview " on page 995 - "Programmatic Descriptions" on page 995 **For API testing:** "Writing Event Handlers for API Test Steps" on page 1935

Context menu options are described below:

> **Note:** The menu commands described below may not be displayed in the exact order presented here. The options available differ depending where in your document you right-click.

Menu Command	Keyboard Shortcut	Description
Object Properties		Opens the "Object Properties Dialog Box" (described on page 1271). **Note:** This option is only available when right-clicking on an object description.

Menu Command	Keyboard Shortcut	Description
Cut		Removes the currently selected step or text from your action or component.
Copy	CTRL+C	Copies the currently selected step or text.
Paste	CTRL+V	Pastes the currently copied step or text.
Delete		Deletes the selected step or text from your action or component.
Comment	CTRL+M	Adds a comment to the selected line.
Uncomment	CTRL+ SHIFT+M	Removes a comment from the selected line.
Indent	TAB	Indents the current line.
Outdent	Shift+Tab	Removes an indent from the selected line.
Insert Standard Checkpoint (GUI testing only)	F12	Inserts a standard checkpoint step into your document.
Insert Output Value (GUI testing only)	CTRL +F12	Inserts an output value step into your document.
Insert Step > Start Transaction (GUI testing only)		Inserts a StartTransaction step into your action, marking the beginning of the transaction to be timed. (Relevant only if the test includes transactions to be used by LoadRunner or Business Availability Center.)
Insert Step > End Transaction (GUI testing only)		Inserts an EndTransaction step in the action, marking the end of the transaction to be timed. (Relevant only if the test includes transactions to be used by LoadRunner or Business Availability Center.)
Insert Step > Step Generator (GUI testing only)	F7	Opens the "Step Generator Dialog Box""Step Generator Dialog Box" (described on page 983).
Action > Properties (GUI tests only)		Displays the Action Properties dialog box.

Menu Command	Keyboard Shortcut	Description
Action > Insert Call to New (GUI tests only)		Opens the "Insert Call to New Action Dialog Box (GUI Testing) " (described on page 913), which enables you to create a new action and inserts it in the specified location.
Action > Insert Call to Copy (GUI tests only)		Opens the "Select Action Dialog Box " (described on page 915), which enables you to insert a call to an editable copy of an existing action.
Action > Insert Call to Existing (GUI tests only)		Opens the "Select Action Dialog Box " (described on page 915), which enables you to insert a call to an existing reusable action.
Action > Split (GUI tests only)		Separates an action into two sibling actions or into parent-child nested actions.
Associate Function Library with <test name> (GUI tests only)		Associates the selected function library with any test in the current solution.
Go to Definition	Ctrl+ Enter	Navigates directly to the definition of the function that is called in the currently selected step.
Insert/Remove Breakpoint	F9	Sets or clears a breakpoint in the test or component.
Enable/Disable Breakpoint	Ctrl+F9	Enables or disables the currently selected breakpoint.
Run from Step (GUI testing only)		Starts a run session from the selected step.
Debug from Step (GUI testing only)		Runs the test or component from the selected step instead of the start of the test or component.
Run to Step (GUI testing only)		Runs until the current step.
Add to Watch	Ctrl+T	Adds the selected item to the Watch pane.
Select All	Ctrl+A	Selects all steps or text.

Regular Expression User Interface

Relevant for: GUI tests and components and API testing

By default, UFT treats all characters in a regular expression literally, except for the period (.), hyphen (-), asterisk (*), caret (^), brackets ([ ]), parentheses (()), dollar sign ($), vertical line (|), plus sign (+), question mark (?), and backslash (\). When one of these special characters is preceded by a backslash (\), UFT treats it as a literal character.

If you enter a special character in the value box of any dialog box that contains the **Regular Expression** check box, and then you select the **Regular Expression** check box, UFT asks you if you want to add a backslash (\) before each special character. If you click **Yes**, a backslash (\) is added before the special character to instruct UFT to treat the character literally. If you click **No**, UFT treats the special character as a regular expression character.

For GUI tests only: All programmatic description property values are automatically treated as regular expressions. For details on programmatic descriptions, see "Programmatic Descriptions" on page 995.

For details, see:

Regular Expression Characters and Usage Options

Relevant for: GUI tests and components and API testing

This section describes some of the more common options that can be used to create regular expressions:

Using the Backslash Character (\ )

A backslash (\) can serve two purposes. It can be used in conjunction with a special character to indicate that the next character be treated as a literal character. For example, \. would be treated as period (.) instead of a wildcard. Alternatively, if the backslash (\) is used in conjunction with some characters that would otherwise be treated as literal characters, such as the letters n, t, w, or d, the combination indicates a special character. For example, \n stands for the newline character.

If the backslash character is not used for either of these purposes, it is ignored.

For example:

- w matches the character w
- \w is a special character that matches any word character including underscore

- \\ matches the literal character \
- \(matches the literal character (
- one\two matches the string onetwo

For example, if you were looking for a Web site called:

newtours.demoaut.com

the period would be mistaken as an indication of a regular expression. To indicate that the period is not part of a regular expression, you would enter it as follows:

newtours\.demoaut\.com

Matching Any Single Character (.)

A period (.) instructs UFT to search for any single character (except for \n). For example:

welcome.

matches welcomes, welcomed, or welcome followed by a space or any other single character. A series of periods indicates the same number of unspecified characters.

To match any single character including \n, enter:

(.|\n)

For more details on the **( )** regular expression characters, see "Grouping Regular Expressions (( ))" on the next page. For more details on the **|** regular expression character, see "Matching One of Several Regular Expressions (|)" on the next page.

Matching Any Single Character in a List ([xy])

Square brackets instruct UFT to search for any single character within a list of characters. For example, to search for the date 1967, 1968, or 1969, enter:

196[789]

Matching Any Single Character Not in a List ([^xy])

When a caret (^) is the first character inside square brackets, it instructs UFT to match any character in the list except for the ones specified in the string. For example:

[^ab]

matches any character except a or b

Note: The caret has this special meaning only when it is displayed first within the brackets.

Matching Any Single Character within a Range ([x-y])

To match a single character within a range, you can use square brackets ([ ]) with the hyphen (-) character. For instance, to match any year in the 1960s, enter:

196[0-9]

A hyphen does not signify a range if it is displayed as the first or last character within brackets, or after a caret (^).

For example, [-a-z] matches a hyphen or any lowercase letter.

> **Note:** Within brackets, the characters ".", "*", "[" and "\" are literal. For example, [.*] matches . or *. If the right bracket is the first character in the range, it is also literal.

Matching Zero or More Specific Characters (*)

An asterisk (*) instructs UFT to match zero or more occurrences of the preceding character. For example:

ca*r

matches car, caaaaaar, and cr

Matching One or More Specific Characters (+)

A plus sign (+) instructs UFT to match one or more occurrences of the preceding character. For example:

ca+r

matches car and caaaaaar, but not cr

Matching Zero or One Specific Character (?)

A question mark (?) instructs UFT to match zero or one occurrences of the preceding character. For example:

ca?r

matches car and cr, but nothing else

Grouping Regular Expressions (( ))

Parentheses (()) instruct UFT to treat the contained sequence as a unit, just as in mathematics and programming languages.

Using groups is especially useful for delimiting the arguments to an alternation operator (|) or a repetition operator: (* , + , ? , { })

Matching One of Several Regular Expressions (|)

A vertical line (|) instructs UFT to match one of a choice of expressions. For example:

foo|bar

causes UFT to match either foo or bar

fo(o|b)ar

causes UFT to match either fooar or fobar

Matching the Beginning of a Line (^)

A caret (^) instructs UFT to match the expression only at the start of a line, or after a newline character.

For example:

book

matches book within the lines—book, my book, and book list, while

^book

matches book only in the lines—book and book list

Matching the End of a Line ($)

A dollar sign ($) instructs UFT to match the expression only at the end of a line.

For example:

book

matches book within the lines—my book, and book list, while a string that is followed by (\n), (\r), or ($), matches only lines ending in that string. For example:

book$

matches book only in the line—my book

Matching a Newline or Carriage Return Character (\n) or (\r)

\n or \r instruct UFT to match the expression only when followed by a newline or carriage return character.

- \n instructs UFT to match any newline characters.
- \r instructs UFT to match any carriage return characters.

For example:

book

matches book within the lines—my book, and book list, while a string that is followed by (\n) or (\r) matches only lines that are followed by a newline or carriage return character. For example:

book\r

matches book only when book is followed by a carriage return

Matching Any AlphaNumeric Character Including the Underscore (\w)

\w instructs UFT to match any alphanumeric character and the underscore (A-Z, a-z, 0-9, _).

For example:

\w* causes UFT to match zero or more occurrences of the alphanumeric characters—A-Z, a-z, 0-9, and the underscore (_). It matches Ab, r9Cj, or 12_uYLgeu_435.

For example:

\w{3} causes UFT to match 3 occurrences of the alphanumeric characters A-Z, a-z, 0-9, and the underscore (_). It matches Ab4, r9_, or z_M.

Matching Any Non-AlphaNumeric Character (\W)

\W instructs UFT to match any character other than alphanumeric characters and underscores.

For example:

\W

matches &, *, ^, %, $, and #

Matching a Decimal Digit (\d)

\d instructs UFT to match any decimal digit.

For example:

\d

matches 1, 2, 4, and 5

Matching an Integer (\D)

\D instructs UFT to match any whole integer.

For example:

\D

matches 145643, 20, 3426767, 4, and 5

Combining Regular Expression Operators

You can combine regular expression operators in a single expression to achieve the exact search criteria you need.

For example, you can combine the `.' and `*' characters to find zero or more occurrences of any character (except \n).

For example,

start.*

matches start, started, starting, starter

You can use a combination of brackets and an asterisk to limit the search to a combination of non-numeric characters. For example:

[a-zA-Z]*

To match any number between 0 and 1200, you need to match numbers with 1 digit, 2 digits, 3 digits, or 4 digits between 1000-1200.

The regular expression below matches any number between 0 and 1200.

([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)

> **Note:** For a complete list and explanation of supported regular expressions characters, see the Regular Expressions section in the Microsoft VBScript documentation (select **Help > HP Unified Functional Testing Help** to open the UFT Help. Then select **VBScript Reference > VBScript > VBScript User's Guide > Introduction to Regular Expressions**).

Regular Expressions in the Find and Replace Dialog Boxes

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

You can use regular expressions in the **Find text** fields to enhance your search in both the "Find Dialog Box (Document Pane Editor) " (described on page 342) and "Replace Dialog Box" (described on page 344).

You can insert a regular expression into your search string by selecting one from a predefined list, or by entering one manually. For a general understanding of regular expressions in UFT, see "Regular Expressions Overview" on page 318.

For more details about find and replace functionality, see "Searching and Replacing in the Editor " on page 317.

> **Note:** Not all the regular expressions listed for other parts of UFT are supported for the search strings in the Find and Replace dialog boxes.

The following table lists the regular expressions supported for search strings, as well as descriptions of each regular expression. For details, see "Regular Expression Characters and Usage Options" on page 352.

Character	Description	Reference
. (period)	Matches any single character	"Matching Any Single Character (.) " on page 353
* (asterisk)	Matches zero or more specific characters	"Matching Zero or More Specific Characters (*)" on page 354
+ (plus sign)	Matches one or more specific characters	"Matching One or More Specific Characters (+)" on page 354
? (question mark)	Matches zero or one specific character	"Matching Zero or One Specific Character (?)" on page 354
^ (caret)	Matches the beginning of a line	"Matching the Beginning of a Line (^)" on page 354
$ (dollar sign)	Matches the end of a line	"Matching the End of a Line ($)" on page 355

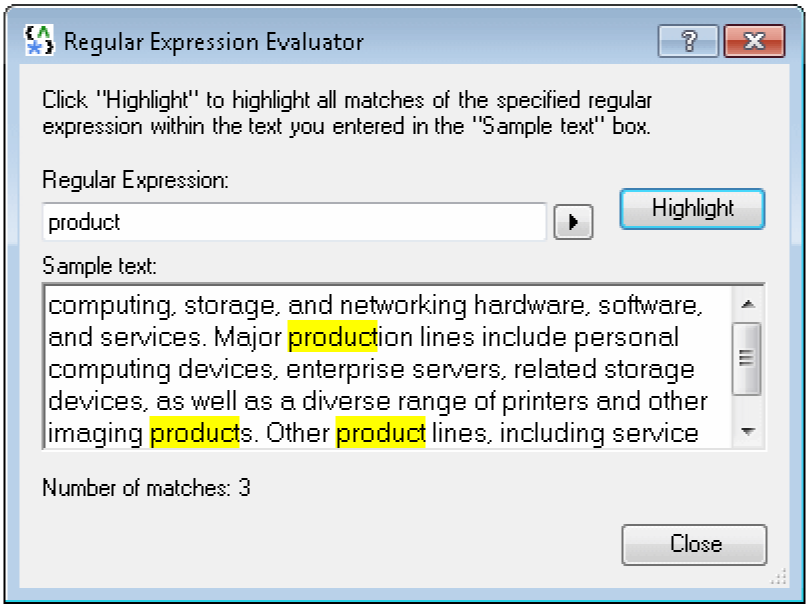
Character	Description	Reference
\n	Matches a line break	"Matching a Newline or Carriage Return Character (\n) or (\r)" on page 355
\r	Matches a carriage return	"Matching a Newline or Carriage Return Character (\n) or (\r)" on page 355
[]	Matches any single character in a list	"Matching Any Single Character in a List ([xy])" on page 353
[^]	Matches any single character not in a list	"Matching Any Single Character Not in a List ([^xy])" on page 353
\w	Matches any alphanumeric character including the underscore	"Matching Any AlphaNumeric Character Including the Underscore (\w)" on page 355
\W	Matches any non-alphanumeric character	"Matching Any Non-AlphaNumeric Character (\W)" on page 356
\d	Matches a decimal digit	"Matching a Decimal Digit (\d)" on page 356
\D	Matches an integer	"Matching an Integer (\D)" on page 356
|	Matches one of several regular expressions	"Matching One of Several Regular Expressions (|)" on page 354
\	Matches a special character treated as a literal character, or a literal character treated as a special character	"Using the Backslash Character (\ ) " on page 352

Regular Expression Evaluator

Relevant for: GUI tests, scripted GUI components, and API testing

This dialog box enables you to create and test a regular expression to determine whether it suits your needs. You can enter a regular expression and sample text to test. When you click **Highlight**, UFT searches the sample text for matches, highlights these matches, and displays the number of matches.

The following example searches for the word product followed by a space or other character because the regular expression includes the period (**.**) character.



Regular Expression Evaluator

Click "Highlight" to highlight all matches of the specified regular expression within the text you entered in the "Sample text" box.

Regular Expression: product

Sample text:

computing, storage, and networking hardware, software, and services. Major production lines include personal computing devices, enterprise servers, related storage devices, as well as a diverse range of printers and other imaging products. Other product lines, including service

Number of matches: 3

To access	1. Do one of the following: - Ensure that a test, action, or component is in focus in the document pane. - In the Solution Explorer, select a test or component node. 2. Use one of the following: - Select the **Tools > Regular Expression Evaluator** menu command. - Click the **Smart Regular Expression** button to the right of a value box in any dialog box (except the Find and Replace dialog boxes) in which the **Regular Expression** check box is selected. Select the **Open Regular Expression Evaluator** list item.
Important information	The list of selectable regular expression characters available from this dialog box is also available from other UFT dialog boxes that contain a selected **Regular Expression** check box and the **Smart Regular Expression** button . **Note:** The Regular Expression Evaluator is not supported for regular expressions in the Find and Replace dialog boxes.
See also	**Conceptual overview:** "Regular Expression Characters and Usage Options" on page 352 **Additional related topics:** "Smart Regular Expression List" on page 361

User interface elements are described below:

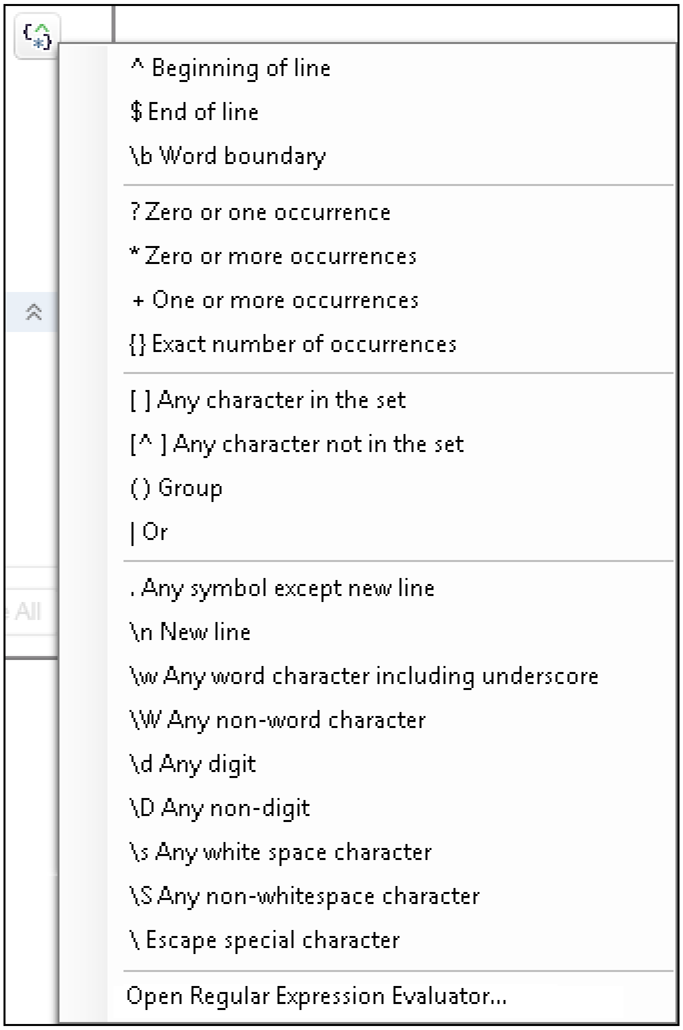
UI Elements	Description
▶	**Smart Regular Expression.** Click to display a list of regular expression characters that you can select. For details on each of the regular expression characters, see "Smart Regular Expression List" on the next page.
Regular Expression	The regular expression to test. Enter the regular expression in this box. **Tip:** Click the right arrow to display a list of regular expression characters that you can insert.
Highlight	Searches for the regular expression in the **Sample text** area and highlights all matches. **Note:** Before you click **Highlight**, make sure that the text to search is displayed in the **Sample text** area.
Sample text	The text to use when testing the regular expression. **Note:** Text may already be displayed in the **Sample text** area when the dialog box opens, depending on the steps you performed to open this dialog box. The **Sample text** area is editable. You can insert any text to test your regular expression.
Number of matches	The number of matches for the regular expression within the sample text. These matches are highlighted in the **Sample text** area.

Smart Regular Expression List

Relevant for: GUI tests and scripted GUI components

This list:

- Displays a list of commonly used regular expression characters.
- Enables you to select a regular expression character from the list and insert it in a value.
- Enables you to access the "Regular Expression Evaluator" (described on page 358).


^ Beginning of line
$ End of line
\b Word boundary
? Zero or one occurrence
* Zero or more occurrences
+ One or more occurrences
{} Exact number of occurrences
[ ] Any character in the set
[^] Any character not in the set
() Group
| Or
. Any symbol except new line
\n New line
\w Any word character including underscore
\W Any non-word character
\d Any digit
\D Any non-digit
\s Any white space character
\S Any non-whitespace character
\ Escape special character
Open Regular Expression Evaluator...

Note: If you open the Smart Regular Expression list from the Regular Expression Evaluator, the **Open Regular Expression Evaluator** command is not included in the list.

To access	Click the **Expression Builder** button to the right of a value box in which regular expressions can be used, such as the Find dialog box.
See also	**Conceptual overview**: "Regular Expressions Overview" on page 318 and "Regular Expression Characters and Usage Options" on page 352 **Additional related topics**: "Regular Expression Evaluator" on page 358

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

UI Elements	Description
<Regular expression options>	The list of regular expression characters that you can select when creating a regular expression. When you select a regular expression character from the list, UFT inserts it in the currently open dialog box and closes the list. Depending on its type, a regular expression character can be inserted: • At the cursor location • Around selected text • In place of selected text **Example 1:** If you select text and then select the **[ ] Any character in the set** option, the brackets are inserted on either side of the selected text, as follows: [text] **Example 2:** If you select text and then select the **\d Any digit** option, the selected content is replaced with **\d**. For example, if you select 1 in Document1 and then select the **\d Any digit** option, the selected content is replaced with **\d**, as follows: Document1 becomes Document\d **Note:** The **\n New line** option is available only if applicable. For details on working with regular expression characters, see "Regular Expression Characters and Usage Options" on page 352.
Open Regular Expression Evaluator	Opens the "Regular Expression Evaluator" dialog box, enabling you to test your regular expression to make sure that it suits your needs. For details, see "Regular Expression Evaluator" on page 358. **Note:** This command is not available from the Smart Regular Expression List dialog box.

Troubleshooting and Limitations - Statement Completion

Relevant for: GUI actions, scripted GUI components, and function libraries

This section describes troubleshooting and limitations for the document pane.

Statement completion is not available for the following types of code:

- **Variables.** For example:

  ```
  Set x = CreateObject(“Application.Excel”)
  x
  ```

- **Class methods.** For example:

  ```
  class fooClass
  publicfunctionfoo
  sin(45)
  end function
  End Class
  Set x = New fooClass
  x
  ```

- **Data tables and checkpoints.**

Chapter 11: Errors Pane

Relevant for: GUI testing and API testing

This chapter includes:

Concepts

Errors Pane Overview

Relevant for: GUI tests and components, function libraries, and API testing

The Errors pane displays a list of errors generated when opening, working with, saving, and running tests, components, function libraries, and user code files. The Errors pane also displays the error severity levels: Message, Warning, or Error. For user interface details, see "Errors Pane User Interface" on page 374.

The following types of errors can occur:

- "Code syntax errors" below
- "Missing resources (GUI testing only)" below
- "Missing References (API testing only) " on the next page
- "Missing Property Values (API testing only)" on the next page

Code syntax errors

UFT checks for syntax errors whenever you save a test, component, function library, or user code file. If a syntax error is present in your code, the error description is displayed as an Error in the Errors pane.

For GUI testing: You can manually check the syntax of an action, scripted GUI component, or function library at any time by selecting **Design > Check Syntax**. If UFT finds any errors, it displays them in the Errors pane. UFT also checks for syntax errors in your script automatically when switching from the Editor to the Keyword View and vice versa.

You can view a description of each of the VBScript errors in the VBScript Reference.

For API testing: You can view a description of code syntax to help you resolve code errors that are displayed in the Errors pane in the Microsoft C# Reference. For details, see http://msdn.microsoft.com/en-us/library/ms228296(v=vs.90).aspx.

For details, see "How to Manage Errors in the Errors Pane" on page 369.

Missing resources (GUI testing only)

Each time you open a test, component, or application area, UFT verifies that the specified resources are available.

If one or more resources cannot be found, UFT lists them in the Errors pane together with the location in which UFT expected to find these resources. If one of the resources listed in this pane is unavailable during a run session, the run may fail.

> **Note:** If a test or application area is associated with a resource that is located on a password-protected network host, the resource is always listed as missing in the Errors pane unless you

> open the resource prior to opening the test or application area.

The Errors pane enables you to locate or remove missing resources from your test or application area. (You resolve missing resources for GUI components from the application area, with the exception of unmapped repository parameters.)

After you successfully handle a missing resource, it is removed from the pane.

For details, see:

- "Missing Resources Types" on the next page
- "How to Manage Errors in the Errors Pane" on page 369
- "How to Locate a Missing GUI Action" on page 371

Missing References (API testing only)

In the Solution Explorer, each API test contains numerous reference files used in the run session of the test. In addition, you can add additional references to a test. For details on the **References** node in the Solution Explorer, see "References Node" on page 482.

When an API test is run, UFT checks that all necessary reference files are present. If a reference file needed for the run session is not present, then this missing reference is displayed in the Errors pane, along with the location where UFT expected to find the reference.

> **Note:** If a reference file is located on a password-protected network host, the reference is listed as missing in the Errors pane unless you add the resource prior to opening the test.

The Errors pane enables you to find the name of the missing reference file and add it to the test. After you save and run the test again, the error disappears from the Errors pane.

For details on adding references to a test, see "Add Reference Dialog Box" on page 485.

Missing Property Values (API testing only)

When you add certain API test steps to the canvas, you are also required to provide input values and property definitions in the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" of the Properties pane (described on page 417) After you add these steps to the canvas, an error message is displayed in the Errors pane explaining the necessary property value to enter.

You can double-click on the error description in the Errors pane to navigate to the field necessary to resolve the error. After entering the requested values, the error is longer listed in the Errors pane.

For details on input properties for API test steps, see "Standard Activities" on page 1611.

Missing Resources Types

Relevant for: GUI tests and components

UFT reports errors for the following types of missing resources:

- "Missing GUI Actions" below
- "Missing Function Libraries" below
- "Missing Shared Object Repositories" below
- "Missing Recovery Scenarios" on the next page
- "Missing Environment Variable Files" on the next page
- "Unmapped Shared Object Repository Parameter Values" on the next page

Missing GUI Actions

If your test contains a call to one or more actions that cannot be found, UFT lists these actions in the Errors pane.

If your test contains calls to more than one missing action, when you locate the missing action in another test, UFT may identify additional missing actions that are found in the same test. This can occur, for example, if the source test containing the actions that are being called was moved to another folder. In such cases, UFT displays a message box prompting you to map these missing actions as well.

You can instruct UFT to locate these actions simultaneously, or you can handle each call to a missing action individually.

Similarly, if you select to remove an action, and your test contains additional missing actions in the same test, a message box opens, asking whether you want to remove all the actions with the same path.

> **Note:** If a test is opened in read-only format, you cannot view or map its missing actions.

For details, see "How to Locate a Missing GUI Action" on page 371.

Missing Function Libraries

If you choose to remove rather than locate a missing function library, keep in mind that removing this association does not remove calls to those functions from your test or component steps. Therefore, make sure that you handle any calls to functions in removed function libraries.

Missing Shared Object Repositories

You resolve a missing object repository by associating a new object repository with your test or application area, using the Associate Repositories dialog box, when working with tests, or the Object Repositories pane of the Application Area, when working with application areas. The missing object repository is still associated with your test or application area, and still appears in the Errors pane. To remove the missing object repository from the Errors pane and your test or application area, you must use the Remove Repository option of the Associate Repositories dialog box when working with tests, or the Object Repositories pane of the Application Area, when working with application areas.

For more details, see:

- **For tests:** "Associate Repositories Dialog Box" on page 1266
- **For components:** "Object Repositories Pane (Application Area)" on page 2105

Missing Recovery Scenarios

If your test, component, or application area, contains more than one missing recovery scenario, then when you locate a missing scenario in a recovery file, UFT may identify additional missing scenarios in that file. You can instruct UFT to locate these missing recovery scenarios simultaneously, or you can handle each missing scenario individually.

Missing Environment Variable Files

If UFT indicates that an external environment variable file is missing, you can navigate to the file and relink it to your test or component.

Unmapped Shared Object Repository Parameter Values

Every repository parameter used in your test or component must have a specified value. This can be a either a default value that was specified when the parameter was created, or it can be a value that you specify in your test or component. (For details on repository parameters, see "Working with Repository Parameters" on page 1287.)

When you open a test or component that uses an object repository with a repository parameter without a value, UFT indicates this by displaying a **Repository Parameters** item in the Errors pane.

For example, suppose your application contains an edit box whose name property changes depending on a selection made in a previous screen. If you parameterized the value of the name property in the object repository using a repository parameter, but a default value was not defined for the repository parameter, you need to define a value for it. When working with tests, you can map the default value to a Data Table parameter, an environment variable, a random number, or a test or action parameter, and when working with components, you can map it to a local or component parameter. You can also define a constant value for it, and so forth.

Tasks

How to Manage Errors in the Errors Pane

Relevant for: **GUI** tests and components, and API testing

This task describes the different operations that can be performed in the Errors Pane and includes the following steps:

- "Select the errors to display" below
- "Check an action for syntax errors (GUI testing only)" below
- "Locate syntax errors" below
- "Locate a missing resource (GUI testing only)" on the next page
- "Locate a missing reference file (API testing only)" on the next page
- "Locate a missing test step property value (API testing only)" on page 371

Select the errors to display

Use the drop-down list at the left side of the pane to select the errors to display in the pane. You can choose to display all errors in the current solution, or sort by individual documents.

Check an action for syntax errors (GUI testing only)

1. In the document pane, or solution manager select the test or action for which you want to check code syntax.
2. In the Errors pane, select whether to check the syntax in a specific test or in the whole solution.
3. Select **Design > Check Syntax**.

   Any syntax errors in the solution or the selected test are displayed in the Errors pane.

Locate syntax errors

In the Errors pane, do one of the following:

- Double-click the error description.
- Right-click the error description and select **Locate**.
- Select the line containing the syntax error and click the **Locate** button.

In the document pane, the cursor jumps to the source of the syntax error. For user code files, the characters containing syntax error are also underlined.

Locate a missing resource (GUI testing only)

1. Prerequisites:

   - **Determine the cause of the problem:** In the Errors, pane, look at the details of the item you want to resolve and analyze the source of the problem. For example, the resource file may have been renamed, move to a different folder, or deleted accidentally or intentionally.

   - **Determine whether to resolve the problem or remove the association of this resource with your testing asset:** Decide whether the missing resource is still necessary for your test or component.

     If you no longer need this resource file, remove the association in the Solution Explorer or right-click the error description and select **Remove Missing Resource**.

2. In the Errors pane, double click the error description. A dialog box opens, enabling you to locate the resource file:

   - **For a missing action:** "Select Action Dialog Box " on page 915. For full task details, see "How to Locate a Missing GUI Action" on the next page.

   - **For a missing function library:** "Open/New <Document>/<Resource> Dialog Box" on page 156

   - **For a missing shared object repository:** "Associate Repositories Dialog Box" on page 1266

   - **For a missing recovery scenario:** "Add Recovery Scenario Dialog Box" on page 1120

   - **For a missing environmental variable file:** "Open/New <Document>/<Resource> Dialog Box" on page 156

   - **For a missing object repository parameter:** "Associate Repositories Dialog Box" on page 1266

   - **For a missing data table:** "Open/New <Document>/<Resource> Dialog Box" on page 156

   In the selected dialog box, navigate to the missing resource and associate it with your test.

   > **Note:** A missing object repository is still associated with your test or application area, and still appears in the Errors pane. To remove the missing object repository from the Errors pane and your test or application area, you must use the Remove Repository option of the Associate Repositories dialog box, when working with tests, or the Object Repositories pane of the Application Area, when working with application areas.

Locate a missing reference file (API testing only)

1. Locate the source of the missing reference files, as described in the step"Locate syntax errors"

(described above).

The relevant test code file opens and the cursor flashes at the place in which the missing reference file is called during a test run displaying the reference file's name.

2. In the Solution Explorer, right-click the **References** node located under an API test and select **Add Reference**.

3. In the "Add Reference Dialog Box" (described on page 485), navigate to the missing reference file and associate it with your test.

Locate a missing test step property value (API testing only)

1. Locate the source of the missing reference files, as described in the step"Check an action for syntax errors (GUI testing only)" (described above).

   The field requiring the missing property value is highlighted in the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" in the Properties pane (described on 417).

2. Enter the required information for the property value.

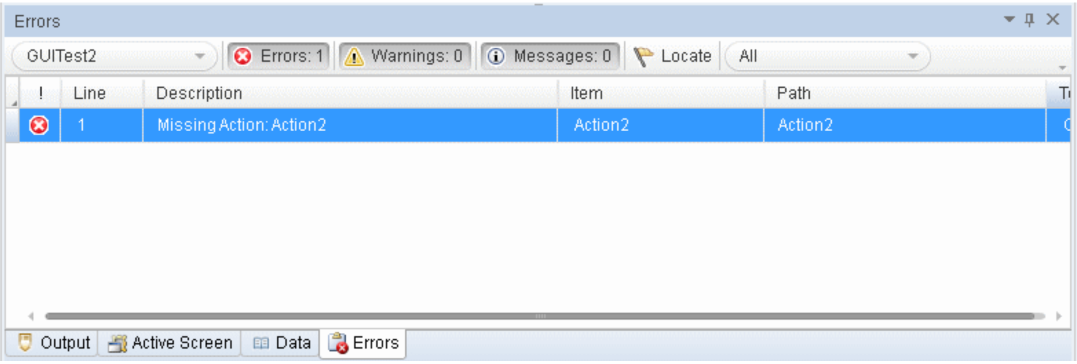
> **Note:** An alert is also displayed in the step in the canvas. For details, see "Alerts (API Testing only)" on page 210.

How to Locate a Missing GUI Action

Relevant for: GUI tests only

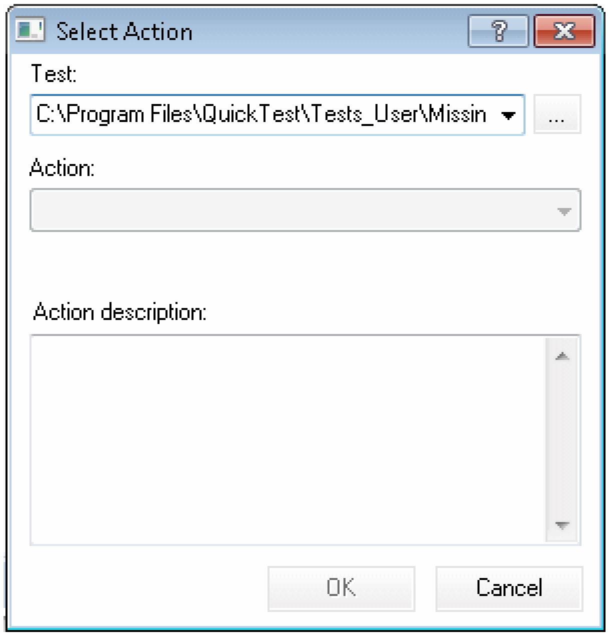
To locate a missing GUI action:

1. In the Errors pane, double-click the action you want to locate or right-click the action and select **Locate** from the context menu.



!	Line	Description	Item	Path
⊗	1	Missing Action: Action2	Action2	Action2

The Select Action dialog box opens.



When the Select Action dialog box opens, the **Test** box displays either the name of the test containing the missing action (if UFT can identify the source test), or **<Current Test>**.

> **Note:** If the missing action is a nested action that is called from another test, you cannot use the **Locate** button to browse to that action. Instead, you must resolve the missing action from within the external test. For example, if ActionAA (in TestA) calls ActionBB (from TestB), and ActionBB calls ActionCC (from TestC), if you open TestA and the call to ActionCC is missing, then you can only resolve the missing action by opening TestB and locating ActionCC. (You cannot resolve it from within TestA.)

2. Click the **Browse** button to find the test that contains the action you want to locate. The **Action** box displays all reusable actions in the test you select.

> **Note:**
>
> - When you select a test, the **Test** box is renamed to **From test**. If the test you select contains reusable actions, these are listed in the **Action** box.
> - You can enter an ALM folder or a relative path in the **Test/From test** box. If you enter a relative path, UFT searches for the test in the folders listed in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** pane). For details, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544 and "Relative Paths in UFT for GUI Testing" on page 130.

> If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, specify an absolute ALM path. For details, see "Relative Paths and ALM" on page 758.

3. In the **Action** list, select the action you want to call. When you select an action, its type (Reusable Action) and description, if one exists, are displayed. This helps you identify the action you want to call. For details on action descriptions, see "General Tab (Action Properties Dialog Box)" on page 900.

4. Click **OK**. UFT updates the test with your changes and removes the action from the Errors pane.

> **Note:** If your test contains additional missing actions that can be located in the same test, UFT opens a message box asking you if you want to map these actions as well. Click **Yes** to map all relevant actions, or click **No** to map only the action you specified.

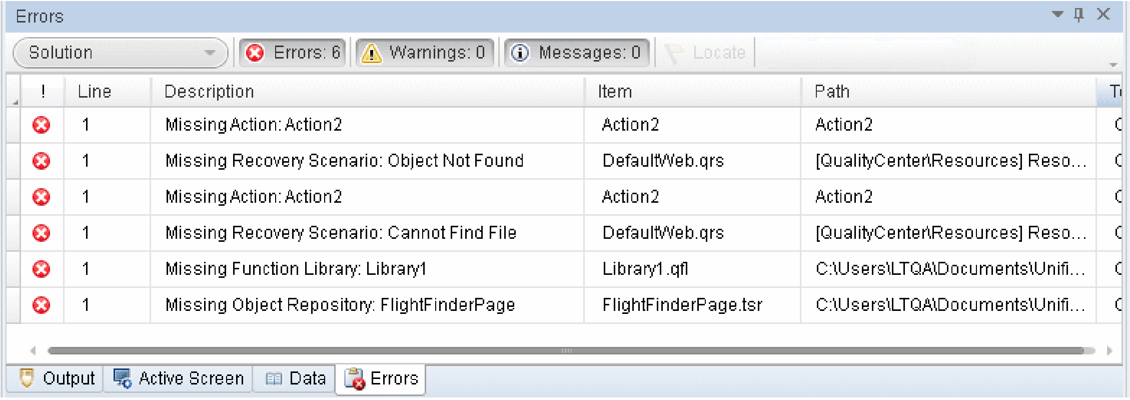
Reference

Errors Pane User Interface

Relevant for: GUI tests and components, function libraries and API testing

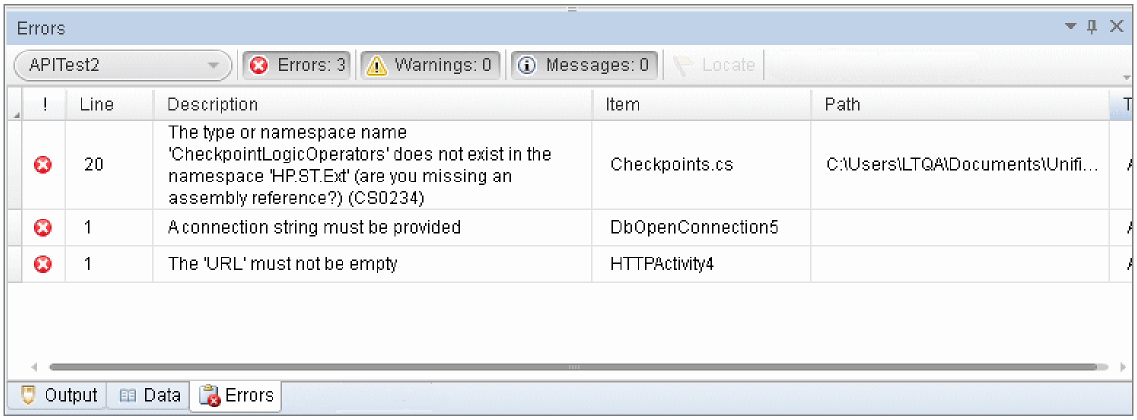
The Errors pane lists the syntax errors found in your test, scripted GUI component, function library, or user code document, and enables you to locate each syntax error so that you can correct it. It also provides a list of the resources that are referenced in your test or component but cannot be found.

The image below shows the Errors pane displayed when opening a GUI test with missing actions, recovery scenarios, function libraries, and object repositories.



!	Line	Description	Item	Path	T
⊗	1	Missing Action: Action2	Action2	Action2	C
⊗	1	Missing Recovery Scenario: Object Not Found	DefaultWeb.qrs	[QualityCenter\Resources] Reso...	C
⊗	1	Missing Action: Action2	Action2	Action2	C
⊗	1	Missing Recovery Scenario: Cannot Find File	DefaultWeb.qrs	[QualityCenter\Resources] Reso...	C
⊗	1	Missing Function Library: Library1	Library1.qfl	C:\Users\LTQA\Documents\Unifi...	C
⊗	1	Missing Object Repository: FlightFinderPage	FlightFinderPage.tsr	C:\Users\LTQA\Documents\Unifi...	C

The image below shows an example of the Errors pane when opening an API test with missing resources and missing property values.



!	Line	Description	Item	Path	T
⊗	20	The type or namespace name 'CheckpointLogicOperators' does not exist in the namespace 'HP.ST.Ext' (are you missing an assembly reference?) (CS0234)	Checkpoints.cs	C:\Users\LTQA\Documents\Unifi...	A
⊗	1	A connection string must be provided	DbOpenConnection5		A
⊗	1	The 'URL' must not be empty	HTTPActivity4		A

To access	Do one of the following: - Select **View > Errors**. - Click the **Errors** tab in the bottom pane.

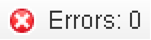
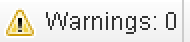
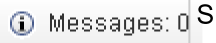
Important information	• The Errors pane displays the errors for all tests contained in your solution. You can also sort to display errors for each individual document contained within the solution. • Each time you open a GUI test or component, UFT automatically checks that all specified resources are accessible. If any resources are not accessible, UFT lists them in the Errors pane. If the Errors pane is not currently displayed, UFT automatically opens it when a missing resource is detected. • Click on a row to jump to the line in the code that generated the error. • Click any column header, for example **Line** or **Item**, to sort by that criteria.
Relevant tasks	• "How to Manage Errors in the Errors Pane" on page 369 • "How to Locate a Missing GUI Action" on page 371 • "How to Run an API Test" on page 643
See also	• "Basic VBScript Syntax" on page 1013 • "Alerts (API Testing only)" on page 210

The following sections describe:

- "Main User Interface Elements" below
- "Context Menu Items" on page 377

Main User Interface Elements

The main user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Show errors from>	Filters the message list by the source of the error. **Default:** Solution
 Errors: 0	Shows or hides syntax errors, property value errors (API testing only) and missing resources.
 Warnings: 0	Shows or hides warnings detected during the run.
 Messages: 0	Shows or hides informational messages detected during the run.

UI Elements	Description
Locate	• **Syntax errors:** Jumps to the line in the Editor that contains the error. • **Missing resources (GUI testing only).** Opens the relevant dialog box for the selected missing resource, enabling you to browse to the correct path for the resource. • **Missing references (API testing only).** Opens the relevant testing code file to highlight the name of the missing resource. • **Missing property value (API testing only).** Highlights the input field in the **Input/Checkpoints** tab in the Properties pane. Available also by right-clicking an error line. For details, see "Context Menu Items" on the next page.
! <Exclamation Point>	**Message type:** • ❌ Error • ⚠ Warning • ℹ Informational message
Line	The line containing the error. For GUI tests, lines are numbered from the beginning of each action or function library. For API tests, the lines are numbered from the beginning of the TestUserCode.cs file. **Note:** For missing resources, the line number is 1.
Description	Description of the error, warning or message and advice on how to fix the problem. For example, a syntax error is displayed if you opened a block of code with a { but did not close it with an }**,** the description is Expected end of statement. **Note:** In certain cases, UFT is unable to identify the exact error and displays a number of possible error conditions, for example: Expected 'End Sub', or 'End Function', or 'End Property'. Check the statement at the specified line to clarify which error is relevant in your case.

UI Elements	Description
Item	• **Syntax errors.** The name of theGUI action, function library, or user code file containing the problematic statement. • **Missing resources (GUI testing only).** The name of the missing resource. • **Missing references (API testing only).** The name of the test user code that contains the call to the missing reference. • **Missing property values (API testing only).** The name of the activity containing the missing property value. • **Missing repository parameter (GUI testing only).** The name and path of the test containing the missing parameter.
Path	The full path of the file that generated the error.
Test	The name of the relevant test or component containing the error.

Context Menu Items

The user interface elements described below are available when you right-click an error in the Errors pane.

UI Elements	Description
Copy	Copies the content of the selected error to the clipboard.
Locate	• **Syntax errors.** Jumps to the location that contains the error. You can double-click a syntax error to locate the error in the relevant document, and then correct it. • **Missing resources (GUI testing only).** Opens the relevant dialog box for the selected missing resource, enabling you to browse to the correct path for the resource. • **Missing references (API testing only).** Opens the test code file that contains the call to the reference file and highlights the name of the missing reference. • **Missing property value (API testing only).** Opens the relevant field in the **Input/Checkpoints** tab in the Properties pane. **Note:** Double-clicking a missing resource item has the same effect as selecting the **Locate** option from the right-click menu or from the toolbar.

UI Elements	Description
Remove **(for missing resources in GUI tests or components only)**	Removes the association of this resource with your action, test, or application area.

Chapter 12: Output Pane

Relevant for: GUI tests and components and API testing

This chapter includes:

Concepts

Output Pane Overview

Relevant for: GUI tests and components and API testing

The Output pane displays the following information:

- Details about assets that cannot be located or loaded during a run session.
- **For GUI tests and components:** Information sent using the **Print** Utility statement during the run session. For details, see "Message Statements" on page 975.
- **For API tests:**Output messages generated in the Output log while compiling and running the test.

For user interface details, see "Output Pane User Interface" on the next page.

Reference

Output Pane User Interface

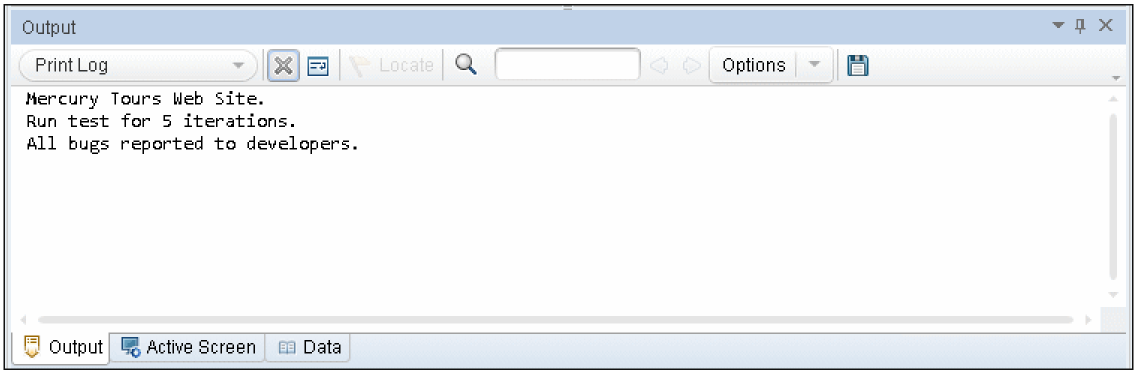
Relevant for: GUI tests and components and API testing

The Output pane displays the following information:

For GUI tests and components: Information sent using the **Print** Utility statement during the run session. For details, see "Message Statements" on page 975.

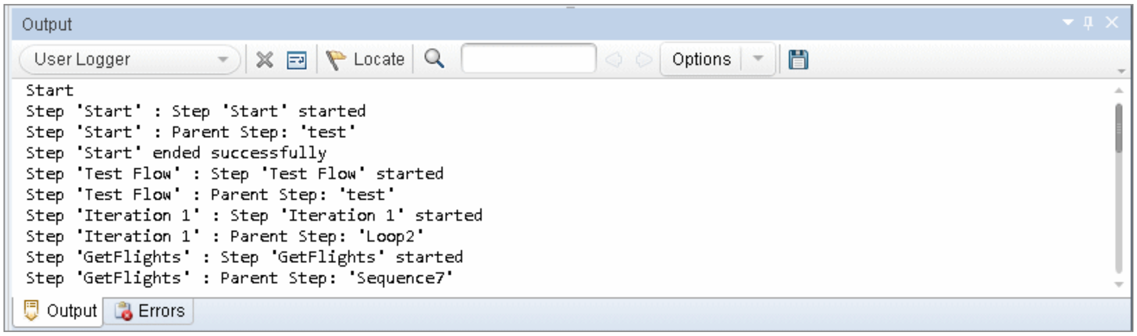
For API tests: Output messages generated in the Output log while compiling and running the test.

The image below shows the Output pane during a GUI test run session.



```
Mercury Tours Web Site.
Run test for 5 iterations.
All bugs reported to developers.
```

The image below shows the Output pane during an API test run session.



```
Start
Step 'Start' : Step 'Start' started
Step 'Start' : Parent Step: 'test'
Step 'Start' ended successfully
Step 'Test Flow' : Step 'Test Flow' started
Step 'Test Flow' : Parent Step: 'test'
Step 'Iteration 1' : Step 'Iteration 1' started
Step 'Iteration 1' : Parent Step: 'Loop2'
Step 'GetFlights' : Step 'GetFlights' started
Step 'GetFlights' : Parent Step: 'Sequence7'
```

To Access	Do one of the following: • Select **View > Output**. • During a paused run session, click the **Output Pane** toolbar button .
Important information	**For API tests:** Click on a row to navigate to the code that generated the output message.

Relevant tasks	**For API tests:** "How to Run an API Test" on page 643
See also	For details on generating output messages during a GUI testing run session, see "Message Statements" on page 975.

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

UI Element	Description
<Show output from>	The type of output to display. The following types are available: - **Build(API testing only)** Displays all API test build information. - **Debug:** Includes debug information, such as all **Print** command (print log) outputs and details about API tests called from GUI tests.
✗	**Clear All Lines.** Clears all of the messages from the message list.
↵	**Toggle Word Wrap.** When selected, wraps the text of each message onto the next line.
Locate	Jumps to the location in the source document relevant to the selected output message. **Note:** This option is relevant for API testing only.
🔍	**<Find box>**.The text string you want to find. You can refine your search by selecting one of the "Options" described below. Press ENTER to begin the search.
← →	**Find Previous / Find Next.** Highlights the next or previous string that matches the text that you entered in the **Find** box. Available only after you enter text in the **Find** box.

UI Element	Description
Options	Enables you to refine your search with the following options: • **Match Case.** Distinguishes between upper-case and lower-case characters in the search. • **Match Whole Word.** Searches for occurrences that are only whole words and not part of longer words. • **Use Regular Expression.** Treats the specified text string as a regular expression. **Note:** Extended regular expressions and multi-line searches are not supported.
💾	**Save Output to a Text File.** Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164), enabling you to save the contents of the message list as a text file.
<Message list>	All messages sent during the test run session. Each message corresponds to one of the following: **For GUI tests and components:** A Print statement line in the action or function library. **For API tests:** A message generated in the Output log.
Copy	Shortcut (right-click) menu option that copies the selected content to the clipboard.
Locate	Shortcut (right-click) menu option that jumps to the relevant line in the source document for the selected output message. **Note:** This option is relevant for API testing only.

Troubleshooting and Limitations - Output Pane

Relevant for: API testing only

- The Output tab may be unable to display the run results for very large tests, exceeding a thousand steps.

Chapter 13: Properties Pane

Relevant for: GUI testing, API testing, and business process testing

This chapter includes:

Concepts

Properties Pane Overview

Relevant for: GUI tests and components, API testing, and business process tests and flows

The properties pane is used for viewing and editing global and specific properties of your tests and components. This pane reflects the active document that is selected in the solution explorer.

For GUI tests and components: The Properties pane displays the properties of your solution, test, action, component, associated function library, application area, object repository, or recovery scenario. It also enables you to edit the parameters of the selected test, action, or component. The available information in this pane corresponds with the available information in the relevant pane of the Settings dialog box or the Properties or Additional Settings panes of an application area.

For user interface details, see "Properties Pane User Interface (GUI Testing)" on the next page.

For details on how to work with test and action input parameters, see "Test and Action Input Parameters" on page 1528.

For API testing: Each step or test activity may contain several customizable properties or parameters. The Properties pane provides several views that allow you to define or assign values to the properties and define handler events.

The Properties pane's toolbar also provides action buttons that allow you to data drive properties and load files.

For details, see "Properties Pane User Interface (API Testing) " on page 402.

For BPT in UFT: When a business process test or flow is selected in the solution explorer, the Properties pane displays basic information about the test and enables you to manage test parameters for the selected test or flow.

When a component or flow is selected in the document pane, the Properties pane enables you to edit parameters for the selected component or flow including promoting parameters to the next level, as well as add or modify other properties..

For details about promoting BPT component and flow parameters, see "Promoting Parameters in a Business Process Test" on page 2123.

For details, see "Properties Pane User Interface (BPT in UFT)" on page 438.

Reference

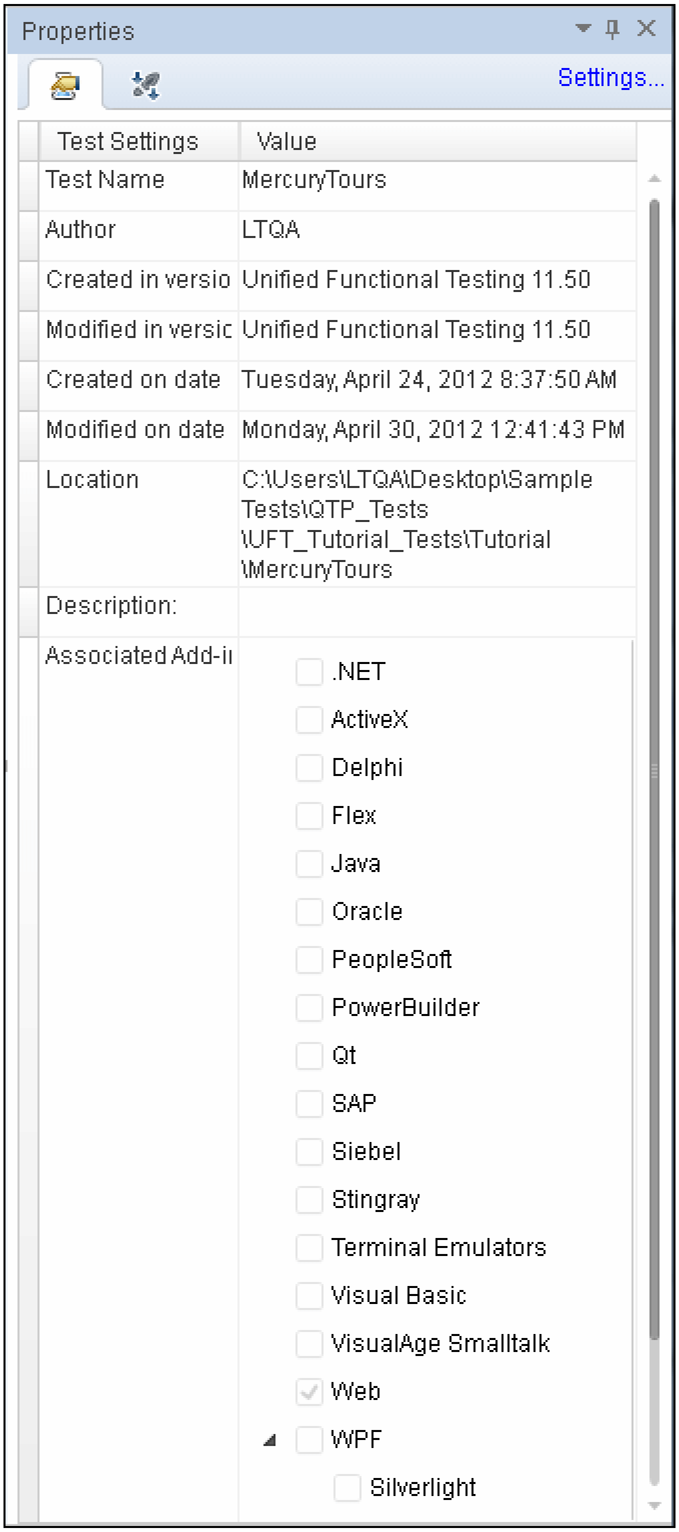
Properties Pane User Interface (GUI Testing)

Relevant for: GUI tests and components

For details on the Properties pane options available for API tests, see "Properties Pane User Interface (API Testing) " on page 402. For details about the Properties pane options available for business process tests, see "Properties Pane User Interface (BPT in UFT)" on page 438

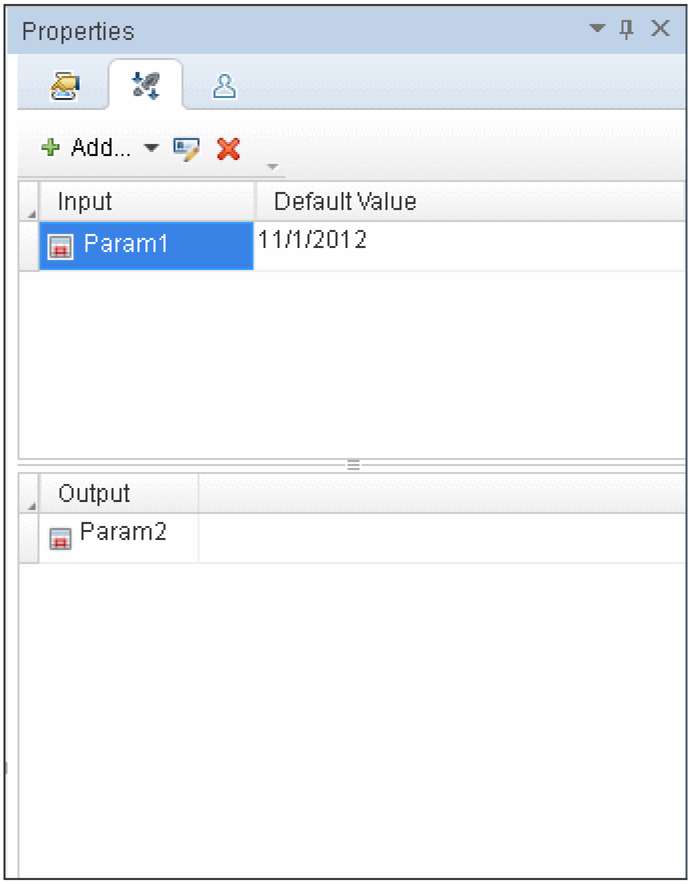
The Properties pane displays the properties and parameters of the selected solution, test, action, component, associated function library, application area, associated object repository, or associated recovery scenario that is selected in the solution explorer. The following tabs are available:

The image below shows an example of the General Properties tab displaying test properties.



Test Settings	Value
Test Name	MercuryTours
Author	LTQA
Created in versio	Unified Functional Testing 11.50
Modified in versic	Unified Functional Testing 11.50
Created on date	Tuesday, April 24, 2012 8:37:50 AM
Modified on date	Monday, April 30, 2012 12:41:43 PM
Location	C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests \UFT_Tutorial_Tests\Tutorial \MercuryTours
Description:	
Associated Add-i	□ .NET □ ActiveX □ Delphi □ Flex □ Java □ Oracle □ PeopleSoft □ PowerBuilder □ Qt □ SAP □ Siebel □ Stingray □ Terminal Emulators □ Visual Basic □ VisualAge Smalltalk ☑ Web □ WPF □ Silverlight

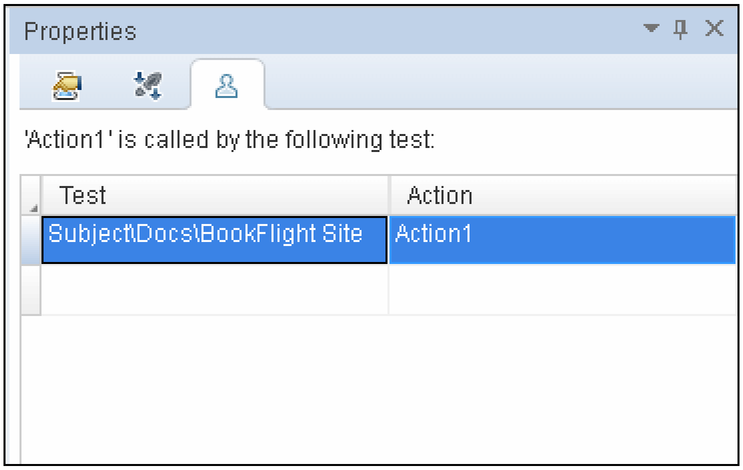
The image below shows the Parameters tab displaying action parameters.



Input	Default Value
Param1	11/1/2012

Output
Param2

This image below shows an example of the Used By tab, which lists the tests and actions containing calls to the actions currently selected in the solution explorer.



'Action1' is called by the following test:

Test	Action
Subject\Docs\BookFlight Site	Action1

To Access	1. Do one of the following: ▪ Ensure that a GUI document is in focus in the document pane. ▪ In the solution explorer, select a GUI document node or one of its child nodes. 2. Select **View > Properties** to open the pane or click the **Properties** pane button.
Important information	• The information displayed in this pane is dependent upon the active document as selected in the solution explorer. • The options that you define in this pane are maintained after you close and reopen the pane, or after you switch between active documents. • The **Undo** command is not supported for this pane. • You can access the Test Settings dialog box by clicking on the **Settings** link in the upper-right corner of the pane when a test is the active document in the solution explorer pane. • Properties are displayed in this pane only for testing documents, not solution items. For example, if a function library is added to your solution but not associated with a test or component, its properties are not visible.
See also	• "Properties Pane Overview" on page 386 • "Properties Pane (Test/Business Component Settings Dialog Box)" on page 590

This section includes the following tabs:

General Properties Tab (Properties Pane - GUI Testing)

Relevant for: GUI tests and components

This tab enables you to view and define general information about your solution, test, action, component, associated function library, associated object repository, or associated recovery scenario. These details correspond with the settings for a test, action, component, or application area in the Properties pane of the Test/Business Component Settings dialog box.

For user interface details relevant for tests and actions, see "Properties Pane (Test/Business Component Settings Dialog Box)" on page 590.

> **Note:** The associated add-ins displayed for a test are read-only. To modify the associated add-ins, see the Properties pane in the Test Settings dialog box.

For user interface details for application areas, see "General Properties Tab (Properties Pane for Application Areas)" below.

> **Note:** The fields displayed in this tab for each type of document selected in the solution explorer vary.

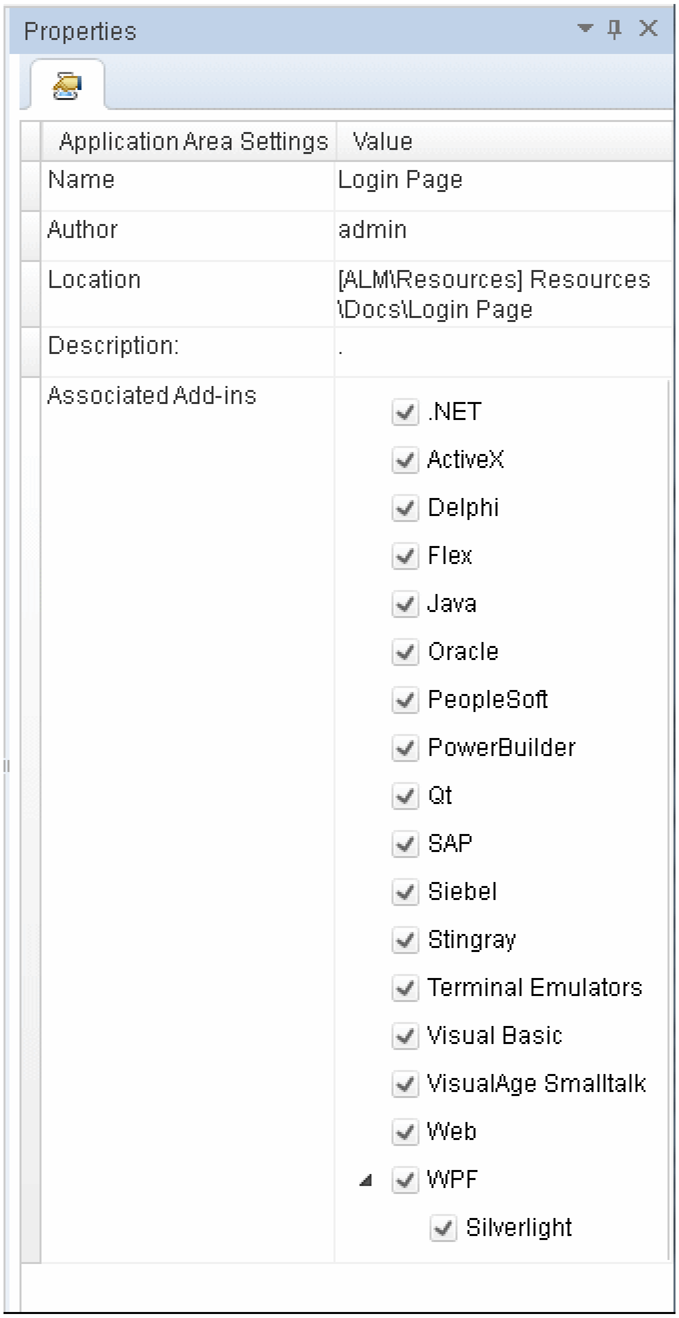
General Properties Tab (Properties Pane for Application Areas)

Relevant for: GUI components only

This pane enables you to view and define general properties for your application area, including its description and any add-ins associated with it.

For details about application areas, see "Application Areas" on page 2095.

The image below shows an example of General Properties tab with application area properties.



Application Area Settings	Value
Name	Login Page
Author	admin
Location	[ALM\Resources] Resources \Docs\Login Page
Description:	.
Associated Add-ins	☑ .NET ☑ ActiveX ☑ Delphi ☑ Flex ☑ Java ☑ Oracle ☑ PeopleSoft ☑ PowerBuilder ☑ Qt ☑ SAP ☑ Siebel ☑ Stingray ☑ Terminal Emulators ☑ Visual Basic ☑ VisualAge Smalltalk ☑ Web ☑ WPF ☑ Silverlight

User interface elements are described below:

To Access	• Do one of the following: ▪ Ensure that an application area is in focus in the document pane. ▪ In the solution explorer, select an application area node. • Select **View > Properties** to open the pane.

Important information	• The information displayed in this pane is dependent upon the active document as selected in the solution explorer. • The options that you define in this pane are maintained after you close and reopen the pane, or after you switch between active documents. • The **Undo** command is not supported for this pane. • Properties are displayed in this pane only for testing documents, not solution items. For example, if a function library is added to your solution but not associated with a test or component, its properties are not visible.
See also	• "Properties Pane Overview" on page 386

User interface elements are described below:

UI Element	Description
Name	The name of the application area (the file name provided when creating the application area).
Author	The ALM user name of the person who created the application area.
Location	The ALM path and file name of the application area.
Description	The description specified for your application area. This field is mandatory, but a non-empty description is created by default for new application areas. It is important that you replace this default with a clear description of the application area. This is because the users who decide which application area to choose when creating a new component, do so based on the **Name** and **Description** of the application area. You can update the description, as needed. For example, if you created an application area but have not finished defining it, you can note this in the **Description** area. Later, after you finalize the application area, you can update the **Description**.

UI Element	Description
Associated add-ins	The add-ins associated with the application area. The associated add-ins are those loaded by UFT when components are accessed. **Note:** When a business process test runs, the add-ins associated with the first component in the test are loaded. Therefore, it is important to ensure that all required UFT add-ins are associated with the application area for the first component in the business process test. For details on associating add-ins, see "Add-in Associations in a GUI Test" on page 583. For details on UFT add-in environments, see the *HP Unified Functional Testing Add-ins Guide*. **Modifying the list of associated add-ins** • If this dialog box contains a child add-in, and you select it, the parent add-in is selected automatically. • If you clear the check box for a parent add-in, the check boxes for its children are also cleared. • If a specific add-in is not currently loaded, but you want to associate it with your test or application area, reopen UFT and load the add-in from the Add-in Manager. If the Add-in Manager dialog box is not displayed when you open UFT, you can choose to display it the next time you open UFT. To do so, select **Display Add-in Manager on startup** from the Startup Options pane of the Options dialog box (**Tools > Options > General** tab **> Startup Options** node). For details, see "UFT Global Options" on page 522. • If you modify this list, you may be required to restart UFT for the changes to take effect.

Parameters Tab (Properties Pane - GUI Testing)

Relevant for: GUI tests and components

This tab enables you to define input parameters that pass values into your test, action, or component and output parameters that pass values from your test, action, or component to external sources. This tab is not available for application areas.

Parameter details in this pane correspond with the parameter settings for a test, action, component as defined in the Parameters pane of the Settings dialog box (**File > Settings**).

User interface elements are described below:

UI Element	Description
Add	Opens the "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)" on page 397, enabling you to add a new input or output parameter.
✎	**Edit Parameter.** Enables you to edit the selected input or output parameter in the "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)" on page 397.
✖	**Delete Parameter.** Deletes the selected input or output parameter. **Caution:** The **Undo** command is not supported for the Properties pane. Therefore, before you delete a parameter, verify that the parameter is not in use. Deleting a parameter that is in use may cause the corresponding step to fail.
🔧	**Maintain SAP Parameters.** Only available when working with SAP Solution Manager in Integration Mode. Allows you to add, delete, and edit Structured Parameters in the SAP Solution Manager Test Data Container. Clicking the icon opens SAP Solution Manager and hides the UFT window.

UI Element	Description
<input parameters list>	The list of input parameters for the selected test, action, or component. The following information is available: - **Name.** The name of the parameter. - **Type.** An icon that represents the type of the parameter. Possible types include: ▪ **String** ▪ **Password** ▪ **Number** ▪ **Date** ▪ **Boolean** ▪ **Any** ▪ **Structured** - **Default Value.** The default value of the parameter. UFT assigns a default value for each type as specified below, if you do not specify a default value for each parameter, and for parameters of components stored in a Quality Center 10.00 or earlier project. ▪ **String.** Empty string ▪ **Password.** Empty string ▪ **Number.**0 ▪ **Date.** The current date ▪ **Boolean.** False ▪ **Any.** Empty string ▪ **Structured.** The default value provided in the SAP Solution Manager Test Data Container **Note:** - The information is displayed in read-only format. To edit any of these fields, use the "Add/Edit Input/Output Parameter Dialog Box (Properties

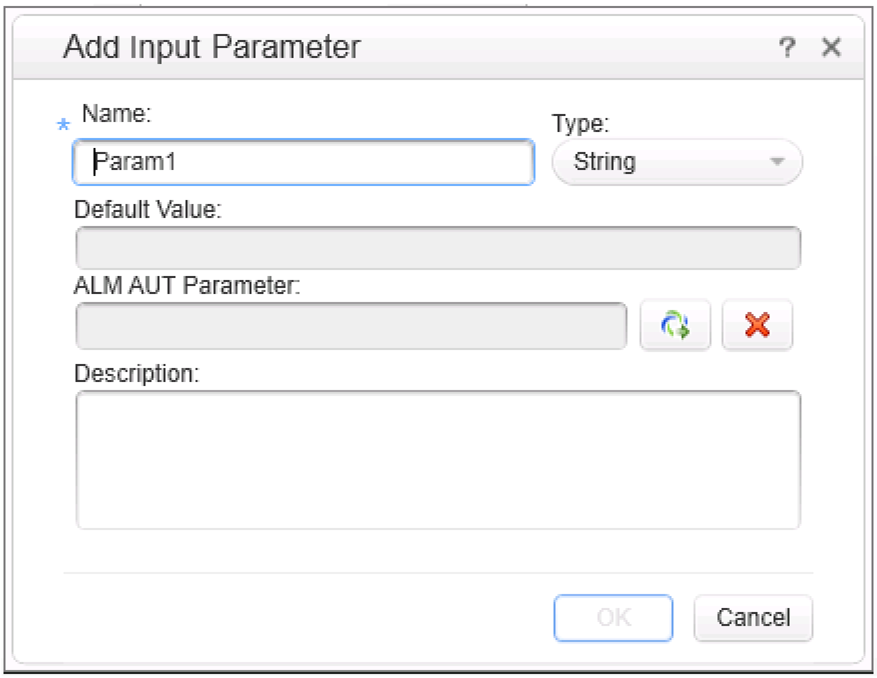
UI Element	Description
	Pane - GUI Testing)" (described on page 397). - An ALM icon next to a parameter's default value indicates that it is an AUT parameter. For details, see "AUT Environment Parameters" on page 713.
<output parameters list>	The list of output parameters for the selected test, action, or component. - **Name.** The name of the parameter. - **Type.** An icon that represents the type of the parameter (as described in the "<input parameters list>"). **Note:** The information is displayed in read-only format. To edit any of these fields, use the "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)" (described on page 397) or double-click the parameter name.
<parameter description area>	The name and description of the selected input or output parameter. **Note:** The information is displayed in read-only format. To edit any of these fields, use the "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)" (described on page 397).

Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)

Relevant for: GUI tests and components

This dialog box enables you to define an input or output parameter to use in the selected test, action, or component.

The image below shows the dialog used to add an input parameter. This example also shows the options available when adding a test parameter to a test stored on an ALM server with Lab Management.



To Access	1. In the solution explorer, select a GUI test, component, or action. 2. In the Properties Pane, select the **Parameters** tab and then do one of the following: ▪ **To add a new input parameter:** Select **Add > Add Input Parameter**. ▪ **To add a new output parameter:** Select **> Add > Add Output Parameter**. ▪ **To edit an existing input or output parameter:** Select an existing input or output parameter and click the **Edit Parameter** button.
Relevant tasks	"How to Run a Test Using Server-Side Execution" on page 727.
See also	• "Properties Pane User Interface (GUI Testing)" on page 387 • "Properties Pane Overview" on page 386 • "Running Tests in Server-Side Execution" on page 713 • "Select AUT Parameter Dialog Box" on page 742

User interface elements are described below:

UI Element	Description
Name	The parameter name (case-sensitive).

UI Element	Description
Type	The parameter type. (Relevant for tests and for components stored in Quality Center 10.00 or earlier) The following types are available: • **String** • **Boolean** • **Date** • **Number** • **Password** • **Any** (tests only) **Note:** For parameters linked to ALM AUT parameters, the type will always be String.

UI Element	Description
Default Value (available only for input parameters)	**For tests:** The default value for the parameter, used during the run session if no other value is provided for the parameter. **Note:** If you want to use fractions in your parameter values, the Decimal symbol character must be a period (.). This is defined in the computer's Region and Language settings. **For components:** An empty string. UFT assigns a default value as specified below, if you do not specify a default value for a test parameter, and for parameters of components stored in a Quality Center 10.00 or earlier project: • **String:** Empty string • **Boolean:** True • **Date:** The current date • **Number:** 0 • **Password:** Empty string • **Any:** Empty string (tests only) When a test or component runs, the actual values used for parameters are generally those sent by the application (UFT or ALM) calling the test or component, as described below: • UFT**.** Input Parameters tab of the Run dialog box. For details, see "Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)" on page 655. • ALM**.** ▪ **For tests:** Test Lab module. For details, see the *HP Application Lifecycle Management User Guide*. ▪ **For components:** Iterations dialog box (Test Plan module). For details, see the *HP Business Process Testing User Guide*. • For business process tests, flows, and components managed in UFT. The Data Pane. For details, see "Data Pane User Interface (BPT in UFT)" on page 265. **Note:** For ALM AUT parameters, you can set this value by selecting **Copy default value (override)** in the "Select AUT Parameter Dialog Box"

UI Element	Description
	(described on page 742).
ALM AUT Parameter	The ALM AUT Parameter from the **Lab Resources > AUT Environment** module in ALM Lab Management. **Note:** Available only if UFT is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
↻	**Select ALM application parameters.** Opens the Select AUT Parameter dialog box for selecting a parameter from an AUT environment defined in the **Lab Resources > AUT Environment** module. For details, see "Select AUT Parameter Dialog Box" on page 742. **Note:** Available only if UFT is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
✖	**Remove link to ALM application parameters.** Removes the linking of the selected parameter from ALM. This parameter will now be an ordinary test property. **Note:** Available only if UFT is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
Description	A meaningful description for the parameter, for example, the purpose of the parameter.

Used By Tab (Properties Pane - GUI Testing)

Relevant for: GUI actions only

This tab displays information on the tests calling the current actions. This tab is visible only when an action is selected in the solution explorer. This tab only displays information for actions that are stored in a test in an ALM project.

> **Note:** In order for the information to be displayed in this tab, the test path must be included in the **Folders** pane in the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** pane). For details, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544.

User interface details correspond with the **Used By** tab in the "Action Properties Dialog Box" (described on page 900).

> **Note:** When you are working with ALM and UFT in different locales, a test name written in an different language than the language of the ALM server is not displayed correctly in this tab.

Properties Pane User Interface (API Testing)

For details about the GUI testing Properties Pane, see "Properties Pane User Interface (GUI Testing)" on page 387. For details about the Properties pane options available for business process tests, see API tests, see "Properties Pane User Interface (BPT in UFT)" on page 438

Relevant for: API testing only

The Properties pane displays a series of tabs showing the properties, schemas, and events for the step selected in the canvas.

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. Select **View > Properties** or click on the Properties pane tab.
Important information	• Only active when the canvas is visible. • The toolbar displays different buttons based on the type of activity or view. • The top row of buttons are **View** buttons, such as **General** or **Events** views. • The bottom row of buttons are buttons that perform actions, such as **Data-Drive** and **Load XML**.
Relevant tasks	• "How to Create an API Test" on page 1598 • "How to Set Security for a Web Service on the Port Level" on page 1885

This section includes:

Properties Pane Tabs

Relevant for: API testing only

The following section describes each of the Properties pane tabs.

Asynchronous Tab (Properties Pane - API Testing)

This tab enables you to indicate whether the response for a Web Service is asynchronous.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **Asynchronous** tab .
Relevant tasks	"How to Create an API Test for an Asynchronous Web Service" on page 1931
See also	"Asynchronous Services" on page 1928

The **Asynchronous** tab's user interface elements are described below.

UI Element	Description
This is an asynchronous call	Indicates whether the call is asynchronous and enables you to specify a listener.
Listen for response on	The port upon which to listen for a response. This property is visible only if you enable the above option.

Attachments Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to view input and output attachments associated with the current test step.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **Attachments** tab .
Relevant tasks	"Add input attachments to the test - optional" on page 1600

The **Attachments** tab's user interface elements are described below (unlabeled elements are shown in angle brackets).

UI Elements	Description
<Input Attachment>	Input attachments for the current Web Service step: - **Type.** Attachment type: NONE, DIME, or MIME. - **Attachments.** A list of the input attachments and their properties: - **Origin.** The name of the file to send as an attachment. Click the **Browse** button to navigate to the directory in which the attachment is saved. **Note:** You can use the output from a different test step as an attachment. Click the **Link to data source** button 🔗 and choose the test property in the "Select Link Source Dialog Box (API Testing)" (described on page 1840). - **Content Type.** The type of file and format. Possible file types include: ○ application/zip ○ image/gif ○ image/ief ○ image/jpeg ○ image/png ○ image/tiff ○ text/plain ○ text/richtext ○ text/html ○ text/xml - **Content ID**. A unique ID for the attachment or **Auto** for an ID generated by UFT. **Note:** To add input attachments, click the ✚ button in the parent node.

UI Elements	Description
<Output Attachment>	The server response saved as an attachment. - **Attachments.** A list of the output attachments and their properties to validate: - **Content.** The file content's checksum value. The checksum is computed by applying the MD5 hash function to the file. - **Content Type.** The attachment's content type. - **Content ID.** The unique ID of the attachment. **Note:** To add output attachments, click the **Attachments** row in the Checkpoints pane, and click ✚ to add an array element. (You may need to expand the column width to access the button). **Tip:** To validate an output attachment, select the check box adjacent to the elements you want to validate. Received attachments are saved in the test's folder, with bin extensions.

Data Sources Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to set the data navigation instructions for the data sources associated with your test flow.

To access	1. Open the Properties pane. 2. Select the Test Flow in the canvas. 3. In the Properties Pane, select the **Data Sources** tab .
Important information	You can have more data sources associated with a test in the Data Pane than are associated with a particular test flow.
Relevant tasks	"How to Set the Data Source Navigation Properties" on page 1825
See also	- "Navigating Within a Data Source" on page 1813 - "Data Navigation Dialog Box" on page 1846

The **Data Sources** tab's user interface elements are described below.

UI Element	Description
Data Navigation Policies	A list of the data sources sorted by: • **Data Source Name.** The name of the data source as it appears in the Data Pane. • **Policy.** A summary of the data navigation policy, for example: **Start at first row, forward one row, wrap around.**
Add	Opens the "Attach Data Source to Loop Dialog Box " (described on page 1839) for selecting data sources from the Data Pane. **Note:** In order to add a define navigation properties for a Data Source, you must associate it with your test in the Data pane. For details, see "How to Add Data Sources to an API Test" on page 233.
Edit	Opens the "Data Navigation Dialog Box" (described on page 1846) for setting the data policy—the way to use the data from the table.
Remove	Deletes the selected data source from the list—not from the Data Pane.

Data Source Properties Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to define the usage properties for your test's data sources.

To access	1. Open the Properties pane. 2. In the Data Pane, select a data source. This tab appears by default and all other properties pane tabs are hidden..
Relevant tasks	"Create a new child relation" on page 1826

The **Data Source Properties** tab's user interface elements are described below.

UI Element	Description
Name	The name of the data source node.

UI Element	Description
Child Relations	A list of the data sources sorted by: • **Child Data Source.** The sheet or table to use as a data source. • **Primary Key.** A column in the parent data source to use as a primary key for the child relation. • **Foreign Key.** The column in the child data source to use as a foreign key for the child relation. To add, edit, or remove a child relation, use the corresponding buttons. For details, see the see "Define New/Edit Data Relation Dialog Box" on page 1849.
Allow other tools to override the data	Allows other tools or tests to override Excel data. Enabling this option allows you to overwrite the data table values with ALM Test Resources. For details, see "Data Awareness in ALM" on page 715. This option also allows action data to be edited when called by another test. For details, see "Data Pane User Interface (API Testing)" on page 252. **Note:** This only applies to Excel data sources. **Caution:** For Excel data sources with multiple sheets, you must enable this option for each data sheet if you did not enable it when importing the Excel file.
Data source policy	The policy by which to handle the data (only relevant for steps that call actions from an external test, to which data was assigned). • **Use data stored with original action (read-only)**. Use the data that is associated with the action, as is. • **Use a local, editable copy**. Create of local copy of the action's data, to allow you to edit the values in the Data Pane. **Note:** This only applies to Excel data sources.

Database Data Source Properties Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to view the properties for your database data sources.

To access	1. Open the Properties pane. 2. Select a Database data source in the Data pane.
Relevant tasks	"Add a database data source" on page 235
See also	"Add New Database Data Source Wizard" on page 258

The **Database Data Source Properties** tab's user interface elements are described below.

UI Element	Description
Name	The name of the data source (read-only).
Connection type	The type of connection: **OleDB** or **ODBC**.
Connection string area	The **Connection string** section contains the following elements: • **Connection string.** A drop down list of connection strings that were defined. • **Build Connection String** . ▪ For **OleDB** types, it opens Microsoft's Data Links Properties dialog box. ▪ For **ODBC** types, it opens the "Select Data Source Name Dialog Box", described on page 262. • **Edit** **.** Opens a text editor for editing the connection string. • **<Connection string text>.** A read-only window showing the selected connection string. • **<Connection status>.** A string indicating whether there is an open connection to the database.

UI Element	Description
SQL statement pane	The **SQL statement** section contains the following elements: - **SQL statement.** A drop down list of SQL statements strings that were defined in this test. - **Build Query Statement.** Opens the **Query Designer**. For details, see the "Query Designer Dialog Box" on page 1724. - **Edit** . Opens a text editor for editing the SQL statement. - **<SQL statement text>.** A read-only window showing the selected SQL statement. - **<Query status>.** A string indicating whether or not the query was executed successfully.

Dependencies Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab shows a list of all external resources (Web Services, REST services, and .NET assemblies) used in your test flow.

To access	1. Open the Properties pane. 2. Select the **Start** or **End** steps in the canvas. 3. In the Properties Pane, select the **Dependencies** tab .

The **Dependencies** tab's user interface elements are described below.

UI Element	Description
Resource Name	The name of the resource as imported into the test
Type	The type of resource, for example, WSDL.
Location	The location in the test in which the resource is used.

Events Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to enter custom code to run special custom code in conjunction with the steps of your test.

To access	1. Open the Properties pane. 2. Select a test step in the canvas. 3. In the Properties Pane, select the **Events** tab ⚡ .
Relevant tasks	"How to Open a Window for Writing Custom Code" on page 1948
See also	"Writing Code for API Test Events - Overview" on page 1938

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

UI Element	Description
<Events list> - default	A list of events for the activity: - **CodeCheckPointEvent.** Triggered when the activity is executed. - **AfterExecuteStepEvent.** Triggered after the activity was executed. - **BeforeExecuteStepEvent.** Triggered before the activity is executed. - **ExecuteEvent.** Triggered when the activity in the step is run. **Tip:** For details about an event, select it and select **Create a default handler**. Refer to comments in the template.
<Events list> - Conditional. Loop steps	**Condition.** Defines the conditions under which the code should be run.
<Events list> - HTTP Receiver	- **OnFilter.** The code to execute when the messages are filtered. - **ReceiveRequest.** The code to execute when a request is received. - **SendResponse.** The code to execute when a response is sent.

UI Element	Description
<Events list> - IBM Websphere MQ	- **BeforeCreateQueueManager.** Code to execute before creating a Queue Manager. - **BeforeMQGet.** Code to execute before getting the messages from the MQ queue through browsing. - **AfterMQGet.** Code to execute after getting the messages from the MQ queue through browsing. - **BeforeMQPutMessage.** Code to execute before putting a message from the MQ queue. - **AfterMQPutMessage.** Code to execute after putting a message from the MQ queue. - **BeforeMQGetMessage.** Code to execute before getting a message from the MQ queue. - **AfterMQGetMessage.** Code to execute after getting a message from the MQ queue. - **BeforeMQPublishMessage.** Code to execute before publishing a message to an MQ topic. - **AfterMQPublishMessage.** Code to execute after publishing a message to an MQ topic. - **BeforeMQReceiveMessage.** Code to execute before receiving a message published on an MQ topic. - **AfterMQReceiveMessage.** Code to execute after receiving a message published on an MQ topic. For task details, see "How to Retrieve Messages from an MQ Queue" on page 1635.
<Events list> - Wait	- **TimeoutReached.** The code to execute when the timeout (Input properties) is reached.

Excel File Properties Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to change the Excel file associated with your Excel data sources.

To access	In the Data pane, select the parent node for an Excel data source. The other Properties pane tabs are hidden.
Relevant tasks	Add list of cross references using bullets for multiple entries. Do not use periods.

The user interface elements are described below.

UI Element	Description
Name	The name of the data source as it appears in the Data pane (read-only).
File Path	The current path to the Excel file (read-only).
Change Excel File	Opens the "New/Change Excel Data Source Dialog Box" (described on page 255) which enables you to select a new file for the selected data sources.

Filter Settings Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to set a filter for received messages. This tab is available for **HTTP Receive** steps and operations from Web Service calls imported as server response steps.

To access	1. Open the Properties pane. 2. Select an HTTP Receive or Web Service step in the canvas. 3. In the Properties Pane, select the **Filter** tab ▽ .
Relevant tasks	"How to Create an API Test for an Asynchronous Web Service" on page 1931
See also	"Asynchronous Services" on page 1928

The **Filter Settings** tab's user interface elements are described below.

UI Element	Description
✖	Clears the text in the filter area.
🔗	Opens the "Select Link Source Dialog Box (API Testing)" (described on page 1840), to allow you to link to existing data.
<filter text>	A text area containing the filtering expression. You can enter free text or a regular expression and use the Select Link Source dialog box to create a data expression. **Note:** When filtering request headers, you can only filter requests that contain both a key and value. Headers that have a key, but no value, cannot be filtered.

UI Element	Description
Filter Type	The type of filtering: **Exact match**, **Starts with**, **Ends with**, **Contains**, and **Regular expression**. **Note:** If you specify **Exact Match** and you are trying to match a key-value pair, you must specify both the key and value strings.

General Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab displays general information about the selected test step.

To access	1. Open the Properties pane. 2. Select any step in the canvas. 3. In the Properties Pane, select the **General** tab 📄.

The **General** tab's user interface elements are described below. The properties differ based on the activity type.

For details about activity-specific general properties, see the specify activity in "Standard Activities" on page 1611.

UI Element	Description
Comment	An editable note describing the purpose of the step. UFT also places the comment in the test report.
Properties - default	- **Step ID.** A unique ID for the step. - **Name.** Step name as it should appear in the canvas.
Properties - HP Automated Testing Tools	- **Test path.** The path of the API or GUI test or VuGen script (LoadRunner's Virtual User Generator) to run. - **Action name.** The name of the action in the test (not relevant for Virtual User Generator scripts). - **Description.** A description of the step. - **Result directory.** The path of the results relative to the solution directory (For **Call API Action or Test** only).

HTTP Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to define request and response data for your test steps that perform an HTTP Request to a Web Service or REST service.

To access	1. Open the Properties pane. 2. Select an HTTP Request or REST service step in the canvas. 3. In the Properties Pane, select the **HTTP** tab .
Relevant tasks	"How to a Create a REST Service" on page 1746
See also	"Network Activities" on page 1684

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

	Pane	
UI Element	**Body type**	**Description**
✖	Request/Response: • XML • Text • File • JSON • Post Form (Request only)	**Remove.** Deletes the contents of the: • Body text • Entered path • Selected value in grid
...	Request/Response: • File	**Browse.** Enables you to load a non-XML file containing the request/response body.
🔗	Request/Response: • XML • Text (Request only) • File • JSON	**Link Body.** Opens the "Select Link Source Dialog Box (API Testing)" (described on page 1840) for selecting a data source.

UI Element	Pane Body type	Description
<checkpoint list>	Response pane	A list of the checkpoints with the following information: - **Name.** The name of the checkpoint as it will appear in the results. - **Regular Expression.** A regular expression representing the expected value. - **Validate.** Compares the actual value with the expected value.
<message body>	Request/Response panes - XML - Text - JSON	The body of the request or response loaded through a file, pasted from a clipboard (**Edit > Paste**), or manually entered (Post form).
<post form list>	Request pane: - Post Form	A list of the Post form elements: - **Name.** The name of the element. - **Value.** The element's value.
Clear	Request/Response: - XML	Clears the contents of the **Request/Response Body** areas.
Import Schema	Request/Response: - XML	Imports an .xsd file containing the schema of the request or response.
Load File	Request pane: - Text	Loads a non-.xml file containing the request.
Load JSON	Request/Response: - JSON	Loads a .json file containing the request or response.
Load XML	Request/Response: - XML	Loads an .xml file with the request or response body.
Request Body	Request pane	The format of the request body: **XML**, **Text**, **File**, **JSON,** or **PostForm.**

	Pane	
UI Element	**Body type**	**Description**
Response Body	Response pane	The format of the response body: **XML**, **Text**, **File,** or **JSON.**

HTTP Receiver Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to define response data for steps that receive a response from a HTTP or SOAP-based Web Service.

To access	1. Open the Properties pane. 2. Select an HTTP Receiver step in the canvas. 3. In the Properties Pane, select the **HTTP Receiver** tab .
See also	"Network Activities" on page 1684

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

UI Elements	Response Body type	Description
Received Message Body	**XML, Text, JSON**	The way in which to represent the response: **XML, Text**, or **JSON.**
Name	**Text**	The text element in the received message.
<regular expression>	**Text**	A grid with regular expression containing the response body values to validate.
Validate	**Text**	Instructs UFT to validate the selected text from the received message.
Clear	**XML, JSON**	Clears the contents of the body.
Import Schema	**XML**	Imports an .xsd file containing the schema of the response.
Load XML	**XML**	Loads an .xml file with the response values.
Load JSON	**JSON**	Loads a .json file with the response values.
<body of received message>	**XML, JSON**	The body of the response loaded through a file.
✖	**Text**	**Clear.** Clears the contents of the selected cells in regular expression grid.

Parameters/Checkpoints Tab (Properties Pane - API Testing)

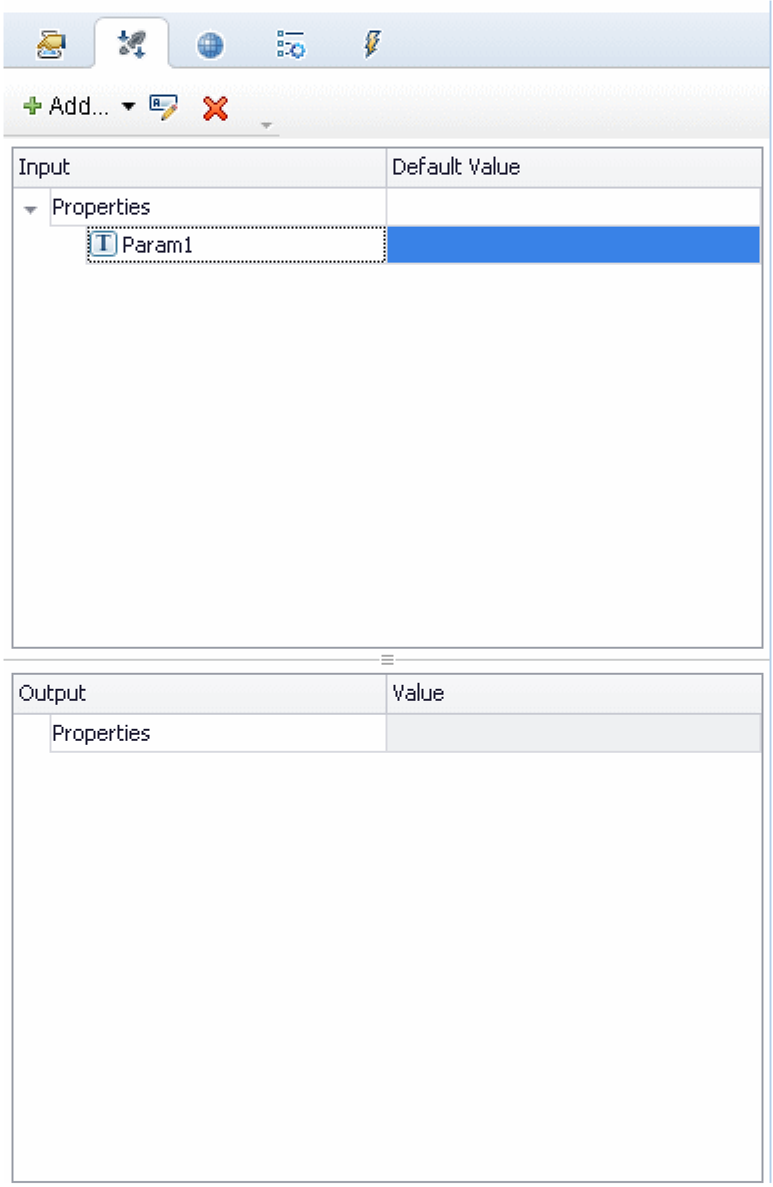
Relevant for: API testing only

This tab enables you to define input and output properties and parameters and checkpoint properties for your test steps.

To access	1. Open the Properties pane. 2. Select one of the following in the canvas. ▪ The **Start** or **End** steps ▪ The **Test Flow** ▪ A test step 3. In the Properties Pane, select the tab. This tab's name and content depend on the test step you selected.
Important information	The name of this tab depends on the test step you select: • If you selected a **Start** or **End** steps, this tab is **Test Input/Output Parameters**. • If you selected the test flow, this tab is **Input**. • If you selected any other step, this tab is **Input/Checkpoints**.

Test Input/Output Parameters Tab

When the **Start** or **End** step of your test is selected, this tab is displayed. You can use the tab to create and define values for input and output properties/parameters to which any test step in your test can link.



The tab's user interface elements are described below (unlabeled elements are shown in angle brackets).

UI Elements	Description
<Properties list>	A list of all available properties.
Add	Opens the "Add Input/Output Property/Parameter Dialog Box (API Testing)" (described on page 1775) which enables you to define custom input and output properties or parameters for your test.
✎	**Edit Parameter.** When a property/parameter is selected, opens the "Add Input/Output Property/Parameter Dialog Box (API Testing)" (described on page 1775), enabling you to edit any custom input or output properties/parameters
✖	**Delete.** Deletes the selected property/parameter.

Input Tab

When the Test Flow is selected, this tab is displayed. You can use the tab to specify properties for iteration test runs.

The following options are available:

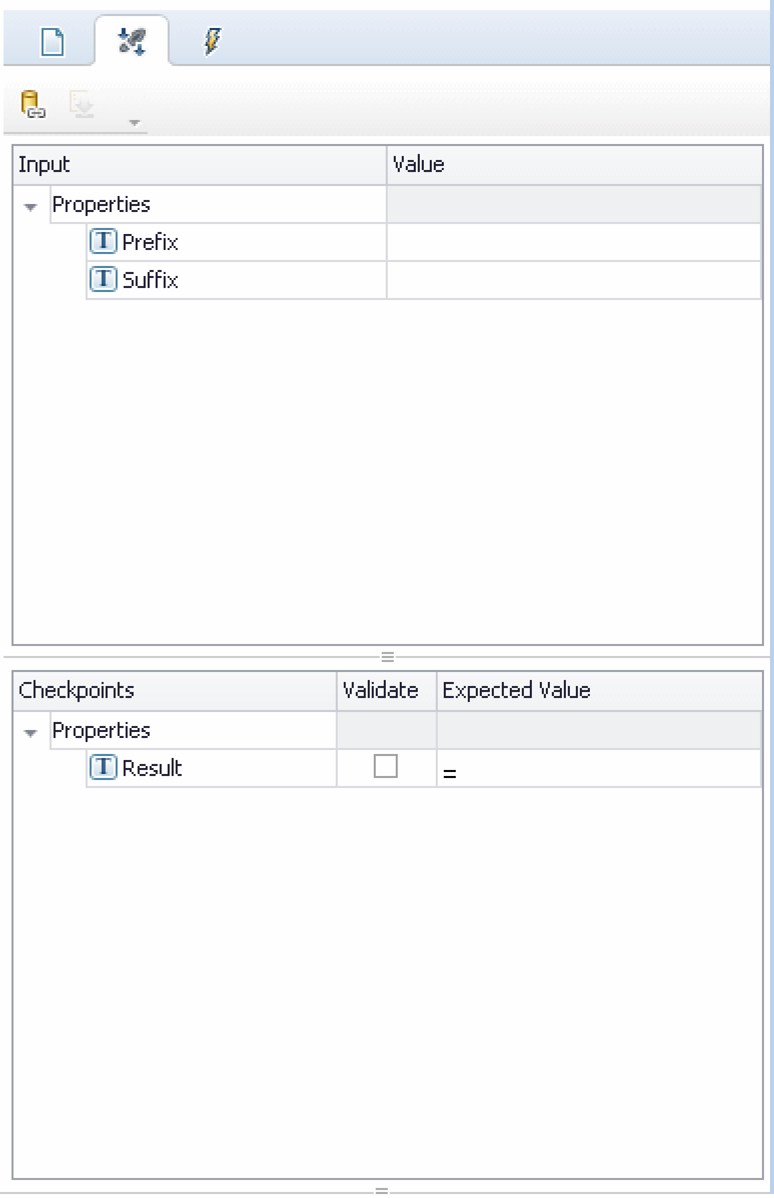
UI Element	Description
Do While Loop - Use condition	Repeats the loop as long as the specified condition is met: • **Variable.** The variable to evaluate. Use the Select Link Source button 🔗 to enter a data source expression. • **Operator.** A comparison operator relevant to the variable such as **=**, **!=**, **Contains**, **Starts** and **Regex**. • **Value.** The value with which to compare the result.
Do While Loop - Use event	Performs iterations until the event returns True. This mode is useful for complex conditions that can be defined in an event handler. When you select this option, the Input Properties grid opens. • Click ➕ to define properties. • Click ⚡ to open the **Events** view. Click **Create a default handler** in the **Handler** column. • Modify or add code to the event handler method that defines your condition.

UI Element	Description
For Each Loop	Performs an iteration for each element in the associated array or collection of objects. Data is selected using the **Select Link Source** button 🔗. **Note:** When you delete data from a data table, it continues to run an iteration for that row, with empty values. To remove an entire row of a data, click the row and select **Delete** from the shortcut menu (only with Excel installations).
For Loop	Performs the Test Flow or custom loop the number of times that you specify in the **Number of Iterations** box.

Input/Checkpoints Tab

When a test step is selected, this tab is displayed. You can use the tab to specify input and checkpoint properties for the selected test step.

For details on the different toolbar buttons that are displayed when a test activity is selected, see "Action Buttons" on page 435



Input	Value
Properties	
Prefix	
Suffix	

Checkpoints	Validate	Expected Value
Properties		
Result	□	=

The tab's user interface elements are described below (unlabeled elements are shown in angle brackets).

UI Elements	Description
<properties list>	A list of all available properties. The fields differ per step type: • For most activities: The activity's **Input** properties. For details on the available input properties, see the individual activity descriptions in "Standard Activity Types" on page 1650. • For Web Service, REST Service, and WADL methods: ▪ the properties as designed in the WSDL or WADL file (for Web Service or Web Application methods), ▪ The custom input or output properties defined for your REST method in the REST Service editor. For details on creating input or output properties for REST Service methods, see "Define custom properties - optional" on page 1748.
<checkpoint list>	A list of the step's output properties/parameters, displaying the following columns: • **Checkpoints.** A list of the output properties/parameters. • **Validate.** When selected, UFT validates the current output properties/parameters when running the step. When cleared, UFT does not check the output properties/parameters. • **Expected Value.** The expected value for the output parameter. Select the down arrow to view a drop down list of comparison operators, number scrolling, and boolean values. You can also manually enter the expected value into the cell. Additional checkpoint properties include: • **Trim whitespace (from start and end of string).** Removes whitespace before and after the text. • **Ignore case.** When looking for a match, ignores the case. • **Stop test if checkpoint fails.** Exits the test run if the checkpoint fails. You set this individually for each checkpoint. **Note:** You must click on a row in the Checkpoint section to see these properties. The first two properties are available only for string data types.

UI Elements	Description
<checkpoints list> (available for- Web Service and SOAP Request steps)	Checkpoints options for Web Service and SOAP request steps include: • **Send request to service.** Sends the step's request to the service when running the test. This is the normal behavior for Web Services and most types of actions. It is enabled by default. **Note:** To send messages using JMS transport, clear this check box. Link a subsequent JMS step to the output of this step. For Web services, you can set the security settings and include attachments, and send this over JMS. • **Validate Structure.** Adds a checkpoint that verifies that the service is in conformance with the schema defined for the SOAP XML response—either for the response defined in the operation or a SOAP Fault response. The Run Results Viewer indicates whether or not the validation succeeded. This option is available only when the **Send request to server** option is enabled. It is enabled by default. **Note:** If you modify the elements in the response, this will not affect the response schema. For example, adding array elements, selecting specific Choice elements, changing the derived type, and so forth, affect only the response—not the schema. To validate the element values, specify expected values in the **Value** column or use an XPath expression. • **Validate WS-I.** Adds a checkpoint to verify that the SOAP response in compliance with WS-I standards. The Run Results Viewer indicates whether the response was in compliance. It also provides a **View Report** link that opens the WS-I Validation report in a separate window.

UI Elements	Description
<XPath checkpoint options> (available for steps with XML output properties)	For steps with XML output properties (such as **String to XML**), the following additional options are relevant: - **XML tab.** A grid representation of the response's schema. In this section you can enter the expected response values. Use the following options to enter your XML: ▪ **Import Schema.** Imports a schema for the XML response. ▪ **Load XML.** Loads an XML file as a basis for the schema of the response. ▪ **Clear.** Removes the displayed schema. - **XPath tab.** A list of XPath expressions used to evaluate the XML response. Use the following options to use XPath expressions: ▪ ✚ Adds a line for a new XPath expression. ▪ ✖ Removes the selected XPath expression. ▪ **Ignore namespaces:** Ignores namespaces in the XPath validation, allowing you to use simple XPath expressions. For details, see "How to Set XPath Checkpoints for Test Steps" on page 1617.

Context Menu Options

The following context menu options are available in this tab:

UI Elements	Description
<context menu>- input properties	Provides shortcuts for including properties and setting their values. • **Collapse/Expand All.** Displays or hides the list of array elements. • **Set Auto-value.** Inserts a sample value for the argument, based on its data type. • **Include/Include All.** Includes the current or all Choice input properties in the test run. • **Exclude/Exclude All.** Excludes the current or all Choice input properties from the test run. • **Copy XPath.** Copies a simplified XPath expression to the clipboard. For details, see "XPath Checkpoints" on page 1613. • **Copy Fully Qualified XPath.** Copies the complete XPath expression of the value to the clipboard. • **Link to Data Source.** Opens the "Select Link Source Dialog Box (API Testing)" on page 1840 • **Clear Cell Contents.** Clears the contents of the cell. **Note:** Some options are available only for specific property types.

UI Elements	Description
<context menu> - checkpoints list	Provides shortcuts for including properties and setting their values. • **Remove Array Element.** Removes the selected array element. • **Duplicate Array Element.** Duplicates the selected array element with its values. • **Select All.** Selects/clears the Validate check box for all child array elements. • **Expand/Expand All.** Expands all elements included in an array. • **Collapse/Collapse All.** Controls the display of array elements. • **Clear/Clear All.** Clears the selected or all check boxes for validation. • **Set Auto-value.** Inserts a sample value for the argument, based on its data type. • **Copy XPath.** Copies a simplified XPath expression to the clipboard. For details, see "XPath Checkpoints" on page 1613. • **Copy Fully Qualified XPath.** Copies the complete XPath expression of the value to the clipboard. • **Link to Data Source.** Opens the "Select Link Source Dialog Box (API Testing)" (described on page 1840). • **Display Outgoing Links.** Lists the property's outgoing links. For details, see "Outgoing Links" on page 1808. • **Insert Keyword.** Insert a keyword in the **Expected Value** column. The available keywords are #NIL#, #EXISTS#, #NOT_FOUND#, and #SKIP#. For details, see "Data Keywords" on page 1817.

Multipart Tab (Properties Pane - API Testing)

Relevant for: API testing only

For HTTP steps, this tab enables you to configure multipart requests. Multipart messages are sent with the HTTP request and consist of two or more header/body sets.

To access	1. Open the Properties pane. 2. Select an HTTP Request in the canvas. 3. In the Properties Pane, select the **Multipart** tab .
Relevant tasks	"How to Send a Multipart HTTP Request" on page 1626

User interface elements are described below:

UI Elements	Description
Enable Multipart	Enables the **Input/Checkpoints** tab for multipart HTTP requests.

For details about the input and output properties, see "Network Activities" on page 1684.

Result Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you check the output of **String to XML** and **String to JSON** activity steps.

To access	1. Open the Properties pane. 2. Select a **String to XML** or **String to JSON** step in the canvas. 3. In the Properties Pane, select the **Results** tab .
See also	• "XML Activities" on page 1717 • "JSON Activities" on page 1690

User interface details are shown below:

UI Elements	Description
Result	A boolean variable indicating whether the string had the correct structure to be converted into XML or JSON.
Validate	Instructs UFT to check if the expected output is correct.
Expected Value	The expected result of the step: **True**, **False**, **0**, or **1**.

Security Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to configure the security settings for your Web Service activities.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **Security Settings** tab .

Relevant tasks	- "How to Set Security for a Web Service on the Port Level" on page 1885 - "How to Set Security for a Specific Step" on page 1886
See also	- "Setting Security Overview" on page 1878 - "Security Settings for Port <Port_Name> Dialog Box" on page 1900

User interface elements are described below:

UI Element	Description
Use the port's security settings	Enables you to use the security settings from the currently selected step's port. **Note:** If you clear this check box, you must configure the security settings in this tab.
Save	Saves the security settings.
Import	Imports a security settings file (.stss).
<security settings area>	Enables you to set security information for the Web Service activity. For full user interface details, see "Security Settings for Port <Port_Name> Dialog Box" on page 1900.
Edit port's settings	Enabled only when the Use the port's security settings option is selected. Opens the "Security Settings for Port <Port_Name> Dialog Box" (described on page 1900), enabling you to configure the security information used for the Web Service's port.

SOAP Fault Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to set the expected result of an Web Service or SOAP activity.

To access	1. Open the Properties pane. 2. Select a Web Service or SOAP step in the canvas. 3. In the Properties Pane, select the **SOAP Fault** tab .
See also	- "Network Activities" on page 1684 - "Negative Testing of Web Services" on page 1738

User interface elements are described below.

UI Element	Description
Fault is expected	Indicates that a SOAP fault is expected during the test run. This setting confirms that the application did not perform a task that it was not designed to perform.
XML tab	The SOAP envelope, containing the Header and Body of the SOAP message. In this section you enter the expected response. You can define **Any** type elements or values for the following properties: • **faultcode.** A status code indicating that the SOAP request was invalid. • **faultstring.** A response string indicating that the SOAP request was invalid. • **faultactor.** An indication of the source of the fault.
XPath tab	A list of XPATH expressions used to evaluate the SOAP response. • ➕ Adds a line for an XPATH expression. • ✖ Removes the selected entry.

Test Settings Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to set global preferences and properties for your test.

To access	1. Open the Properties pane. 2. Select a step in the canvas. 3. In the Properties Pane, select the **Test Settings** tab .
Important information	Test settings apply to all steps in the test.

User interface elements are described below.

Test Settings Tab - Action Buttons

UI Elements	Description
Export	Exports the current Test settings to an XML file.
Import	Imports an XML file with previously saved Test settings.

Load Settings

UI Elements	Description
Load Enabled (read-only)	Indicates whether the test is enabled for load testing. To enable this capability, select **Design > Operation > Enable Test for Load Testing**. **Note:** Once a test is enabled for load testing, you cannot disable it.

JVM (Java Virtual Machine) Settings

These settings are relevant for all Java related activities such as JMS and **Call Java Class** activities.

UI Elements (A-Z)	Description
Additional VM Parameters	Extra parameters to send to the JVM such as **Xbootclasspath**, and any parameters specified by the JVM documentation.
Classpath	The vendor implementation of Java classes together with any other required supporting classes, as determined by the implementation vendor.

JMS Settings

UI Elements (A-Z)	Description
Automatically generate selector	Generates a selector for the response message with the correlation ID of the request (**No** by default). Each JMS message sent to the server has a specific ID. Enable this option if you want UFT to automatically create a selector that includes the message ID. **Note:** This option only affects the **Send and Receive Message from Queue** activity.
JMS connection factory	The JNDI name of the JMS connection factory. This setting is unique for each test.
JMS security credentials	The principal's credentials for the authentication scheme.
JMS security principal	Identity of the principal (for example the user) for the authentication scheme.

UI Elements (A-Z)	Description
JNDI initial context factory	The fully qualified class name of the factory class that will create an initial context. It provides a list of context factories and enables manual entries.
JNDI provider URL	The URL of the service provider. For example: Websphere - iiop://myserver:myport
Number of JMS connections per process	The number of JMS connections each execution process creates. The default is 1, and the maximum is 50. The fewer connections you have, the better your performance.
Received message timeout options	The timeout for received messages. • **Indefinite wait.** Wait as long as required for the message before continuing. • **No wait.** Do not wait for the Receive message, and return control to the script immediately. If there was no message in the queue, the operation fails. • **User defined timeout.** Wait a specified number of seconds for the message. If it does not arrive, the operation fails.
User defined timeout	The amount of seconds to wait for the message before timing out. The default is 20 seconds.

.NET Settings

UI Elements	Description
Assembly paths	The .NET assembly paths for the test. This entry should include all relevant folder paths and environment variables. Separate multiple values with semicolons.

General Settings

UI Elements	Description
Stop test on step failure	Exits the test if a step fails during the test run. **Default:** True

Reporting Settings

UI Elements	Description
Use the reporting mechanism	Tells UFT to automatically open the Run Results Viewer after a test run.

Test Variables Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to set global test variables for use in your test.

To access	1. Open the Properties pane. 2. Select the **Start** or **End** step in the canvas. 3. In the Properties Pane, select the **Test Variables** tab 🌐.
Relevant tasks	"How to Define API Test Properties or User/System Variables" on page 141
See also	• "New Test Profile Dialog Box" on page 172 • "Manage Profiles Dialog Box" on page 173

User interface elements are described below.

UI Element	Description
User variables	A list of the user variables for all profiles. When enabling the **Compare Profiles** option, the grid displays the variable values for each profile in separate columns.

UI Element	Description
System variables	A read-only list of common system variables and their values: • ScenarioID • SystemTempDir • GroupName • TestDir • TestName • LocalHostName • OS • OSVersion • ProductDir • ProductName • ProductVer • UserName.

Test Variables Tab - Action Buttons

UI Elements	Description
+	**Add New User Variable.** Adds a new user variable to the all profiles.
✖	**Remove User Variable.** Deletes the selected user variable from all profiles.
▥+	**Add New Profile.** Adds a new User Variable profile to the list.
✎	**Edit Active Profile.** Opens the Manage Profiles dialog box that lets you to remove or rename profiles.
⚖	**Compare Profiles.** Displays the profiles side-by-side for comparison.
<Profile list>	A list showing the existing User Variable profiles.

XML Body Tab (Properties Pane - API Testing)

Relevant for: API testing only

This tab enables you to configure the response XML data for a SOAP Request step.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **XML Body** tab .
See also	"Network Activities" on page 1684

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

UI Elements	Description
Grid	Shows a grid view of the schema.
Text	Opens an editable text view for the schema.
Revert	Text view only—reverts back to the last version of the schema since you last opened the Text view.
⊗	**Syntax error.** Data at the root level is invalid. Move the mouse over the icon to see the error's line number (Text view only).
⚠	**Syntax warning.** A warning indicating a problem in the entered text, for example, if the specified element is not in the namespace. For details, move the mouse over the icon (Text view only).
Import Schema	Imports an .xsd file representing the schema, for both Input properties and checkpoints.
Load XML	Loads an .xml file containing values for the schema, for both Input properties and checkpoints.
✖ Clear	Deletes the contents that you entered for the schema. This affects both the Grid and Text views.
Input pane	A grid or text representation of the schema's request header and body.
Checkpoints pane - XML view	A grid representation of the response's schema. In this section you can enter the expected response values. - **Import Schema.** Imports a schema for the XML response. - **Load XML.** Loads an XML file with the expected response values. - **Clear.** Deletes the contents of the schema.

UI Elements	Description
XPath Checkpoints pane	A list of XPath expressions used to evaluate the XML response. • ➕ Adds a line for a new XPath expression. • ✖ Removes the selected XPath expression. **Ignore namespaces:** Ignores namespaces in the XPath validation, allowing you to use simple XPath expressions. For details, see "How to Set XPath Checkpoints for Test Steps" on page 1617. **Note:** To retrieve an XPath expression, use the right-click menu options in the Grid view.

Action Buttons

Relevant for: API testing only

The following buttons represent actions, primarily performed on step properties. The availability of the buttons depends on the step type.

UI Elements	Description
	Data Drive Entire Step. Data-drives all values for the step. It adds data expressions to the **Value** columns of all input properties. For details, see "How to Data Drive an API Test Step" on page 1827.
	Edit Property. Opens the Edit Property dialog box for changing the name, type, or description of a property. **Note:** Only available for Custom Code, REST method, and Start/End steps.
	Load from Replay. Loads data from a recent replay in order to populate the service's properties with values (for Web Services only). For details, see "How to Run an API Test" on page 643.
✖	**Remove Property.** Deletes the selected property from the list. **Note:** Only available for Custom Code, REST method, and Start/End steps.

UI Elements	Description
	Add Input/Output Property/Parameter. Opens the "Add Input/Output Property/Parameter Dialog Box (API Testing)" (described on page 1775), allowing you to define a new property and its data type. **Note:** Only available for Custom Code, REST method, and Start/End steps, or when selecting the entire canvas. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Load XML	Loads XML data in order to populate a service's schema with values (for Web Services only). For details, see "How to Run an API Test" on page 643.
Load options	Loads an XML file containing the **Compare XML** ignore settings that you saved with another test.
Save options	Saves the **Compare XML** ignore settings so that you can load them into a future test.
Select GUI/API Test	For **Call GUI Test** and **Call API Test** steps: Invokes the "Select Action or Test Dialog Box" (described on page 1797), enabling you to selecting a test to call from this step.
Select Java File	Opens the "Java Class Dialog Box" for setting additional classpaths for the call (for Call Java Class steps only). For details, see the "Java Class Dialog Box" on page 1727.
Refresh GUI/API Test	For **Call GUI Test** and **Call API Test** steps: Updates the test from its original location.
Load JSON	Loads a JSON file to convert to or from a string. **Note:** Only available for **JSON to String** and **String to JSON** steps.

Value Column Icons

Relevant for: API testing only

The following buttons represent information, drop-down lists, or actions within the Properties pane's **Value** column:

UI Elements	Description
◤	**Include argument in the request.** Toggle to clear the triangle and exclude the argument.
NIL	**Set to NIL.** Toggle to clear the icon and remove the NIL value assignment.
🔒	**Read-only node.** Values that cannot be changed such as properties linked to a data source other than a constant value, or nodes with read-only attribute in the activity signature, first HTTP header, and so forth.
⚠	**Warning.** A warning related to the data source. For example, The data types of the link source and link destination do not match.
⌄	- For **Input properties**: A drop-down list of possible values. For example, for boolean values it provides a list of values for boolean type data: true, false, 0, or 1. For date types, it opens a calendar. This drop-down arrow is located adjacent to the **Link to a data source** button 🔗 . - For **Checkpoints**: A drop-down list of the relevant comparison operators such as: =,!=, >, >=, <, <=, Starts, Ends, Contains, or Regex. The greater than operator, >, when used with strings, indicates that it appears later in the alphabet.
⇕	**Scroll.** A scroll control for integer data types.
...	**Browse.** Enables you to locate a file or folder for example, when using a **File System** type step. For an **Open Connection** step, this opens the "Connection Builder Dialog Box" (described on page 1720).
🔗	**Link to a data source.** Opens the "Select Link Source Dialog Box (API Testing)" (described on page 1840), allowing you to select values for the property from a data source.
🔗	**Display list of outgoing links.** Shows a list of all input properties that link to this output property. For details, see "Outgoing Links" on page 1808.

Array Control Buttons

Relevant for: API testing only

The following buttons allow you to handle array type properties. These buttons are adjacent to the property name in the Properties pane's left pane.

After you add array elements, you can set the number of iterations to use the different array values. For details, see "Flow Control Activities" on page 1651.

UI Elements	Description
➕	**Add array element.** Adds one array element to the selected node in the in the **Properties** tree.
✖	**Remove array element.** Removes the selected array element from the **Properties** tree.
⧉	**Duplicate array element.** Adds a copy of the selected array element.

Properties Pane User Interface (BPT in UFT)

For details about the GUI testing Properties Pane, see "Properties Pane User Interface (GUI Testing)" on page 387. For details about the API testing Data Pane, see "Properties Pane User Interface (API Testing) " on page 402.

Relevant for: business process tests and flows

This pane displays properties, parameters, and other details defined for business process tests, flows, components and groups, and enables you to modify many of them. The tabs displayed, and in some cases, the options displayed on each tab, differ depending on whether you are editing a test or flow, and have selected a component, flow, or group instance in the document pane.

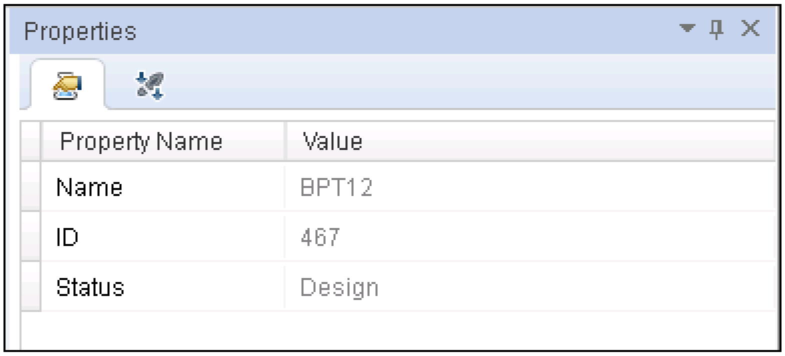
This section includes the following:

General Properties Tab (Properties Pane - BPT)

Relevant for: business process tests and flows

This tab displays read-only information about the business process test or flow selected in the Solution Explorer.

The image below shows an example of the General Properties tab displaying properties for a business process test or flow. The General Properties tab is available only when a business process test or flow is selected in the Solution Explorer.



Property Name	Value
Name	BPT12
ID	467
Status	Design

To access	1. In the Solution Explorer, make sure that the business process test or flow for which you want to view properties is selected. 2. Select **View > Properties** or click the **Properties** pane button . The General Properties tab is displayed by default.
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
Name	The name of the selected test or flow.
ID	The ALM ID of the selected test or flow.
Status	The ALM status for the selected test or flow.

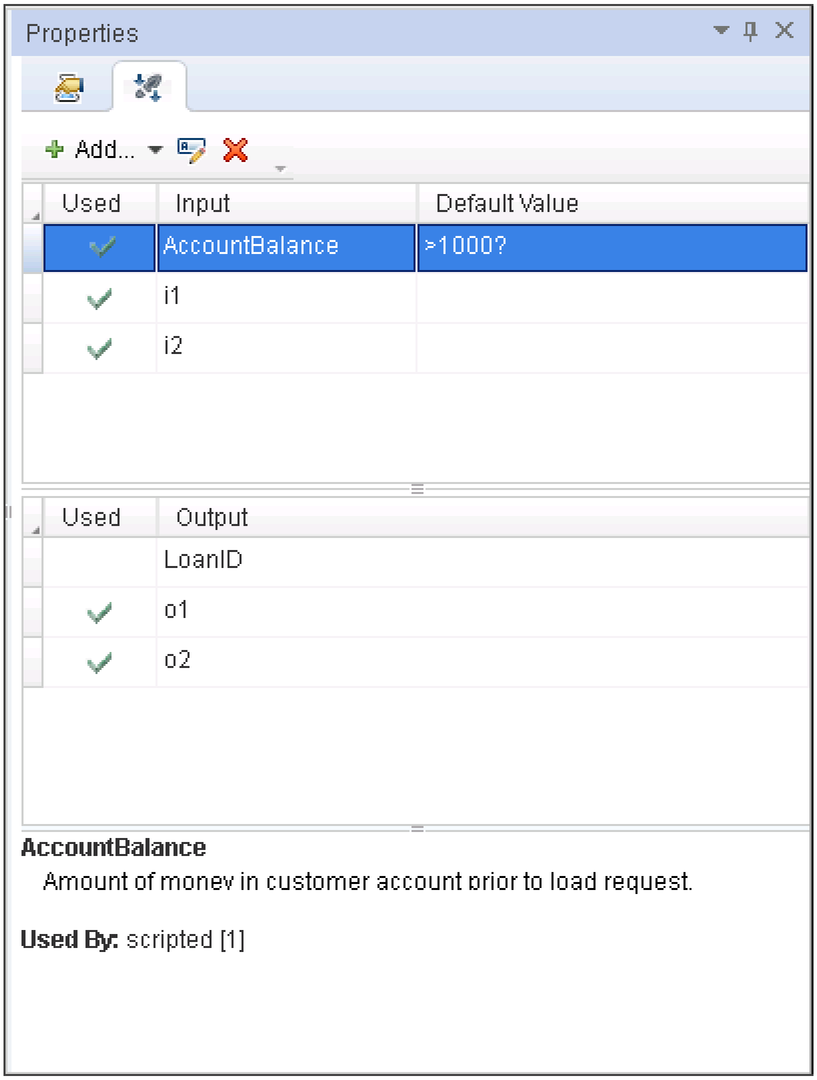
Test Parameters Tab (Properties Pane - BPT)

Relevant for: business process tests and flows

This tab enables you to add, delete, and modify test, or flow input and output parameters.

The image below shows the Test Parameters tab displaying input and output parameters for a flow selected in the Solution Explorer. (Output parameters are not displayed for tests.)

The Test Parameters tab is available only when a business process test or flow is selected in the Solution Explorer.



Used	Input	Default Value
✓	AccountBalance	>1000?
✓	i1	
✓	i2	

Used	Output
	LoanID
✓	o1
✓	o2

AccountBalance
Amount of money in customer account prior to load request.

Used By: scripted [1]

To access	1. In the Solution Explorer, make sure that the business process test or flow for which you want to view properties is selected. 2. Select **View > Properties** or click the **Properties** pane button . 3. Click the Test Parameters tab.
Important information	"Considerations for BPT Test Iterations and Parameters" on page 224 "Promoting Parameters in a Business Process Test" on page 2123
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

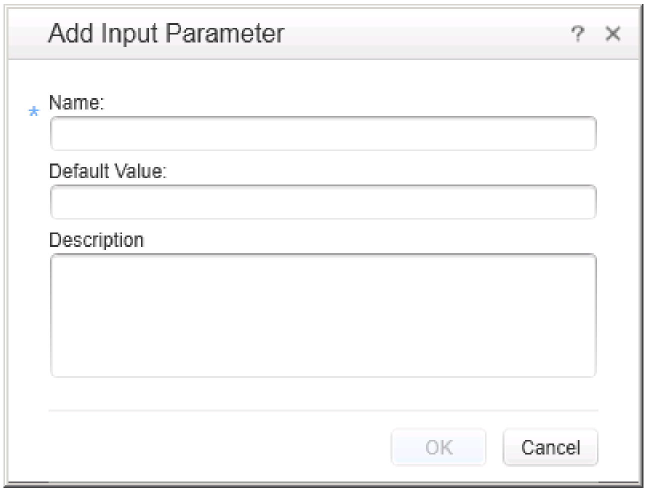
UI Element	Description
Add	Opens the Add/Edit Input/Output Parameter dialog box, enabling you to add a new input or output parameter. For details, see "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - BPT)" on the next page.
✎	**Edit Parameter.** Opens the Add/Edit Input/Output Parameter dialog box, enabling you to edit the selected parameter. For details, see "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - BPT)" on the next page.
✕	**Delete Parameter.** Deletes the selected parameter.
<input parameters list>	The list of input parameters for the test or flow selected in the Solution Explorer. The following information is available: **Used.** Indicates whether the parameter is used in the test or flow. **Input.** The name of the input parameter. **Default Value.** The default value of the input parameter. This value is used during a run session if no other value is supplied by the component.
<output parameters list>	The list of output parameters for the selected flow, and is displayed only for flows. The following information is available: **Used.** Indicates whether the parameter is used in the test or flow. **Output.** The name of the output parameter.

UI Element	Description
<parameter description area>	Lists the following information about the selected parameter: • **Name and description.** These are entered in the "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - BPT)" below. • **Parameter promotion status. (Used By:)** The components or flows that use the selected parameter, if the parameter was promoted from the component or flow in the test.

Add/Edit Input/Output Parameter Dialog Box (Properties Pane - BPT)

Relevant for: business process tests and flows

This dialog box enables you to define an input or output parameter to use in the selected test or flow.



To access	1. In the Solution Explorer, select a business process test or flow. 2. In the Properties Pane, select the **Test Parameters** tab and then do one of the following: ▪ **To add a new input parameter:** Select **Add > Add Input Parameter**. ▪ **To add a new output parameter:** Select **Add > Add Output Parameter**. ▪ **To edit an existing input or output parameter:** Select an existing input or output parameter and click the **Edit Parameter** 📝 button.

Important information	You can add or edit output parameters only when editing a flow. These options are disabled when editing a business process test. "Considerations for BPT Test Iterations and Parameters" on page 224
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
Name	The parameter name (case-sensitive).
Default Value (available only for input parameters)	The default value for the parameter, used during the run session if no other value is provided for the parameter.
Description	A meaningful description for the parameter, for example, the purpose of the parameter.

Parent topic: "Test Parameters Tab (Properties Pane - BPT)" on page 440

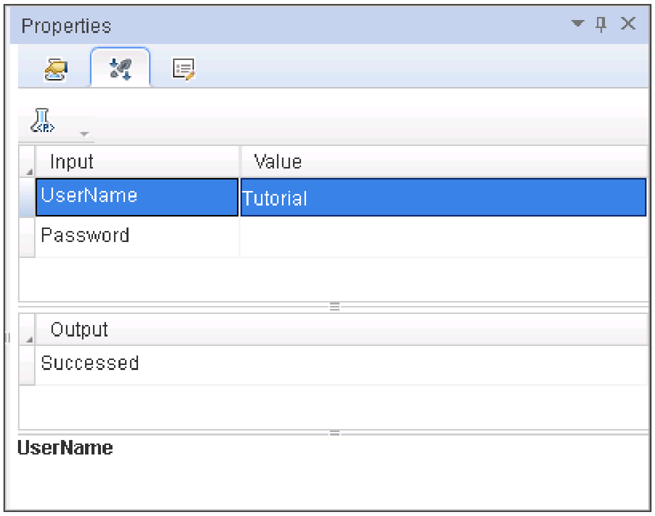
Parameters Tab (Properties Pane - BPT)

Relevant for: business process tests and flows

This tab enables you to define parameter data for a selected component or flow in a business process test or flow.

The Parameters tab is available only when a component or flow is selected in the document pane.

The image below shows the Parameters tab displaying input and output parameters for the selected component or flow. Neither of the input parameters shown in this image have been promoted.

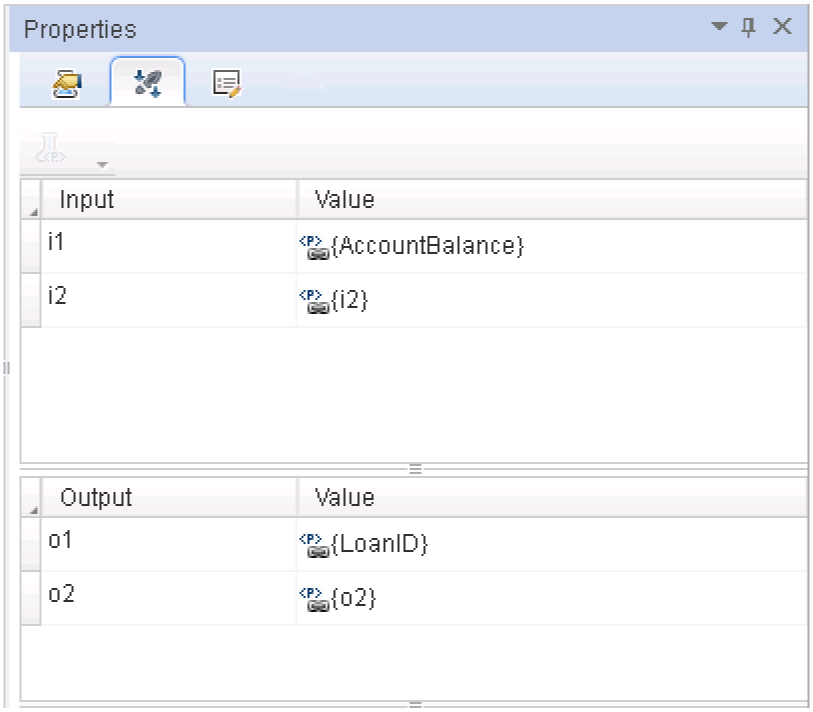


Input	Value
UserName	Tutorial
Password	

Output
Successed

UserName

The image below shows the Parameters tab displaying input and output parameters for the selected component or flow. All of these parameters have been promoted to the next level.



Input	Value
i1	{AccountBalance}
i2	{i2}

Output	Value
o1	{LoanID}
o2	{o2}

To access	1. In the Solution Explorer, make sure that the business process test or flow for which you want to view properties is selected. In the document pane, make sure that specific component or flow you want to modify is selected. 2. Select **View > Properties** or click the **Properties** pane button . The Parameters tab is displayed by default.
Important information	"Considerations for BPT Test Iterations and Parameters" on page 224 "Promoting Parameters in a Business Process Test" on page 2123
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
	Promote parameters. Promotes the selected parameters to the next level, and replaces the selected parameter value with a link to the new parameter. If there are no test parameters with the same name, or if the **Always link to existing test parameters** option is not selected in the BPT Testing tab in the Options dialog box, UFT creates new test parameters, and links the selected component parameter value to the new test parameter values. If test parameters already exist with the same name, and the **Always link to existing test parameters** option is selected, the select component parameter value is automatically linked to the existing test parameter value. This button is disabled if the component or flow parameter is already linked to a parameter at the next level. For more details about promoting parameters, see "Linking Parameters in Business Process Tests" on page 2119. For configuration details, see "BPT Testing Tab (Options Dialog Box)" on page 570.

UI Element	Description
Input	The list of input parameters for the component or flow selected in the document pane. The following information is available: **Input:** The name of the input parameter. **Value:** The parameter's value. Enter text or link the parameter to another test, flow, or component parameter using the **Select Link Source** dialog box (described on page 2128). Hover over the **Value** cell to display the **Link** and **Remove Link** buttons. **Caution:** Removing this link will remove the linked data for the selected parameter in all iterations for the selected component. The **<p> linked** icon indicates that the parameter value is linked to another parameter.
Output	The list of output parameters for the component or flow selected in the document pane. The following information is available: **Output:** The name of the output parameter. **Value:** The parameter's value. Enter text or link the parameter to another test, flow, or component parameter using the **Select Link Source** dialog box (described on page 2128). Values are displayed for output parameters only when a component is in a flow. If you are editing a flow, hover over the **Value** cell to display the **Link** and **Remove Link** buttons. These buttons are not displayed if you are editing a test. **Caution:** Removing this link will remove the linked data for the selected parameter in all iterations for the selected component. The locked icon indicates that the parameter value is linked to another parameter.

Properties Tab (Properties Pane - BPT)

Relevant for: business process tests and flows

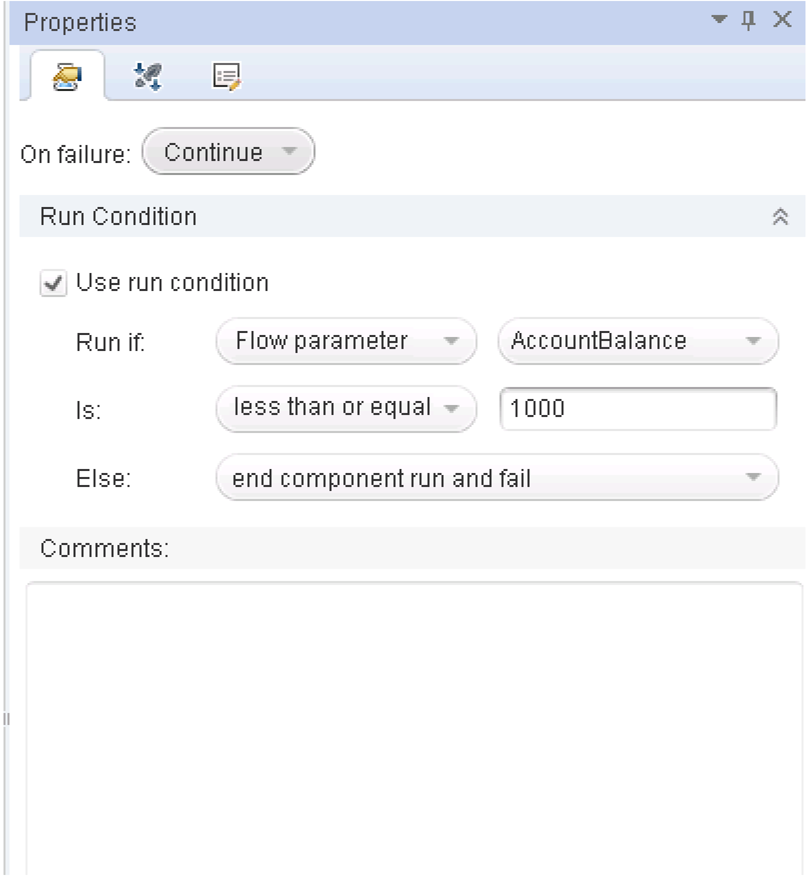
This tab enables you to modify run conditions, On Failure settings, and comments for a selected component instance.

The options displayed in this pane depend on whether you are editing a test or a flow, and have selected a component, flow, or group in the document pane:

- If you are editing a business process test, you can edit only on failure settings and comments for the selected component or flow.
- If you are editing a business process flow, but have selected a group in the document pane, you can only edit comments for the selected group.
- If you are editing a business process flow, and have selected a component, you can modify run conditions, On Failure settings, and comments for the selected component.

  The image below shows an example of the Properties tab, when a component is selected in the document pane, and the user is editing a flow.

  The Properties tab is available only when a component or flow is selected in the document pane and, the content will differ depending on whether you are editing a test or a flow, and what sort of instance you selected in the document pane.



Properties

On failure: Continue

Run Condition

☑ Use run condition

Run if: Flow parameter AccountBalance

Is: less than or equal 1000

Else: end component run and fail

Comments:

To access	1. In the Solution Explorer, make sure that the business process test or flow for which you want to view properties is selected. In the document pane, make sure that specific component or flow you want to modify is selected. 2. Select **View > Properties** or click the **Properties** pane button. 3. Select the Properties tab.
Important information	"Considerations for BPT Test Iterations and Parameters" on page 224
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
Run conditions (for flows, with a component selected)	
Use run condition	Indicates whether to use the defined run conditions for the selected component. Select the checkbox to enable the other run condition options.
Run if:	Enables you to begin a conditional statement, defining that the selected component should run only if it matches the defined conditions. This dropdown menu lists the parameter types defined for the selected component. The following types are available: • **Input parameter.** Available only if one or more input parameters are defined for this component. • **Output parameter.** Available only if one or more output parameters are defined for one or more of the previous components in the flow. • **Flow parameter.** Available only if one or more flow input parameters are defined. **Tip:** When creating a run condition on a parameter that contains a dynamic date value, define the run condition as a static date (for example,12/10/2011), which is compared to the actual date used in the run.

UI Element	Description
<parameter name>	Lists the available parameters. If you selected **Output parameter**, the component name is appended to the parameter name. Encrypted parameters are not listed. **Tip:** Parameters with encrypted values cannot be chosen from the list when defining run conditions. It is recommended that you do not use parameters whose default values are encrypted when defining run conditions.
Is:	Lists the operators that need to be met for the component to run. The following conditions are available: • equal to • not equal to • less than • less than or equal • greater than • greater than or equal After you select one of the options from the dropdown menu on the left, enter the value for the relevant option in the text box on the right.
<value>	Enables you to enter the valid value for the condition.
Else:	Specifies what to do if the condition is not met. The following options are available: • **Skip to next component and continue.** If the condition is not met, the selected component does not run, and the test run continues with the next component in the flow. The component is not displayed in the run results. • **End component run and fail.** If the condition is not met, the component for which the run condition is set does not run, but instead sets the status of the component run as Failed. The flow either continues to the next component or ends, depending on the failure condition set for the component. **Tip:** Your selection from the **Else** dropdown menu applies only if the run condition is not met. To specify whether to continue or end the entire run if a component run fails, modify the **On failure** settings for the component.
On failure settings (for tests and flows)	

UI Element	Description
On failure:	Enables you to define whether a run continues or ends if a specific business component or flow in the test fails. • **Continue.** The business process test will run the next business component or flow if the selected component fails. By default, this failure condition is defined for each component when it is added to a test • **Exit.** The business process test run ends if the selected business component fails.
Comments (for tests and flows, when a component, flow, or group is selected)	
<Comments box>	Enables you to view and edit comments for the selected component or flow. Comments are free text entries that you can use to improve readability. For example: • Add comments before each section of a component's automated tests to specify what that section includes. • Add comments that describe the steps to be included before your application is ready to be tested. When your application is ready, you can use such comments to verify that every item that needs to be tested is included in the steps.

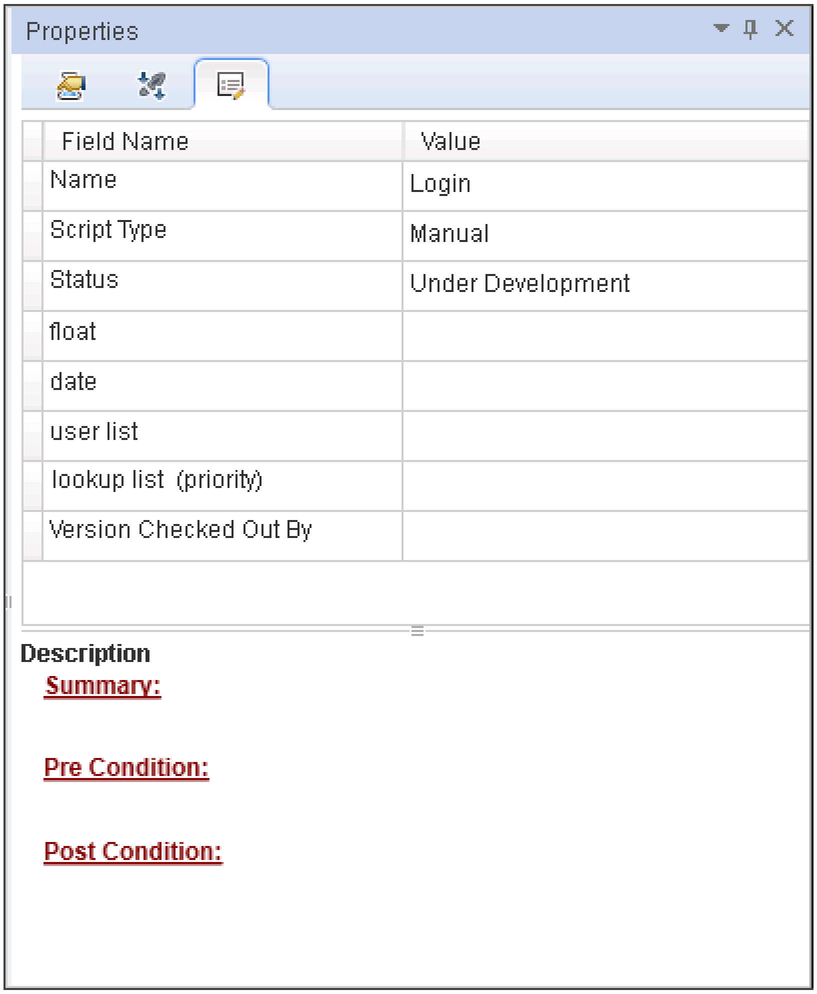
Component Details Tab (Properties Pane - BPT)

Relevant for: business process tests and flows

This tab enables you to view the description defined in ALM for the selected component, as well as fields and their values, including any user defined fields.

The image below shows an example of the Component tab, displaying the description and fields defined in ALM for the selected component.

The Component tab is available only when a component or flow is selected in the document pane.



Field Name	Value
Name	Login
Script Type	Manual
Status	Under Development
float	
date	
user list	
lookup list (priority)	
Version Checked Out By	

Description

Summary:

Pre Condition:

Post Condition:

To access	1. In the Solution Explorer, make sure that the business process test or flow for which you want to view properties is selected. In the document pane, make sure that specific component or flow you want to modify is selected. 2. Select **View > Properties** or click the **Properties** pane button . 3. Select the **Component Details** tab.
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
Field name	The name of the field. The following fields are always displayed: - **Name.** The component's name - **Script Type.** The type of component. - **Status.** The component's ALM status The **Version Checked** field is visible when the component is currently checked out of an ALM project, and defines the user to whom the component is checked out.
Value	The field value, if defined.
Description	The description defined for the selected component or flow.

Chapter 14: Run Step Results Pane

Relevant for: API testing only

This chapter includes:

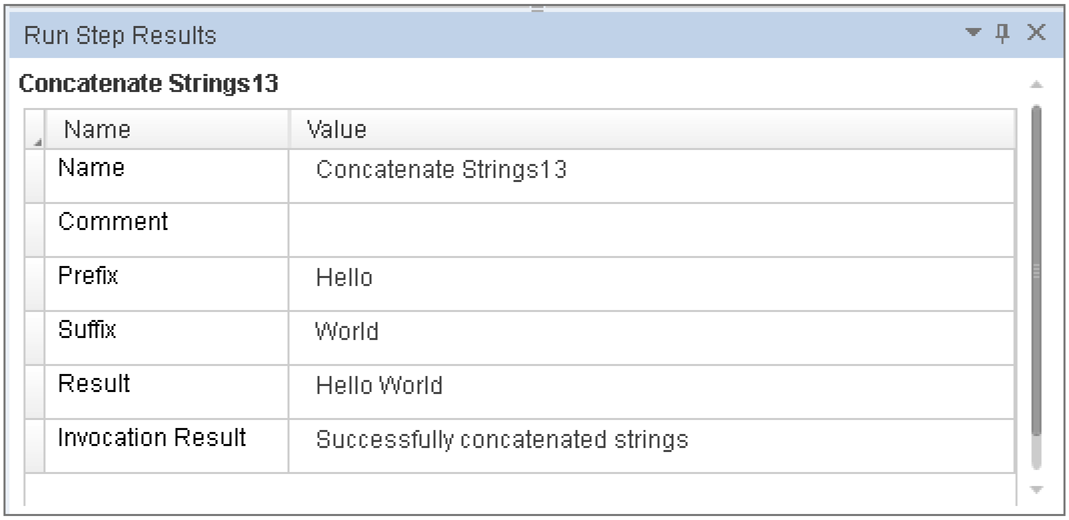
Concepts

Running Test Steps - Overview

Relevant for: API Tests

Before you run your test, you can test individual steps to make sure they run properly. This feature, Run Step, is available directly from the canvas and is available for most step types. To run an individual step, select it and choose **Run Step** from its right-click menu.

After you run the step, the Run Step Results pane opens and shows the properties and results.

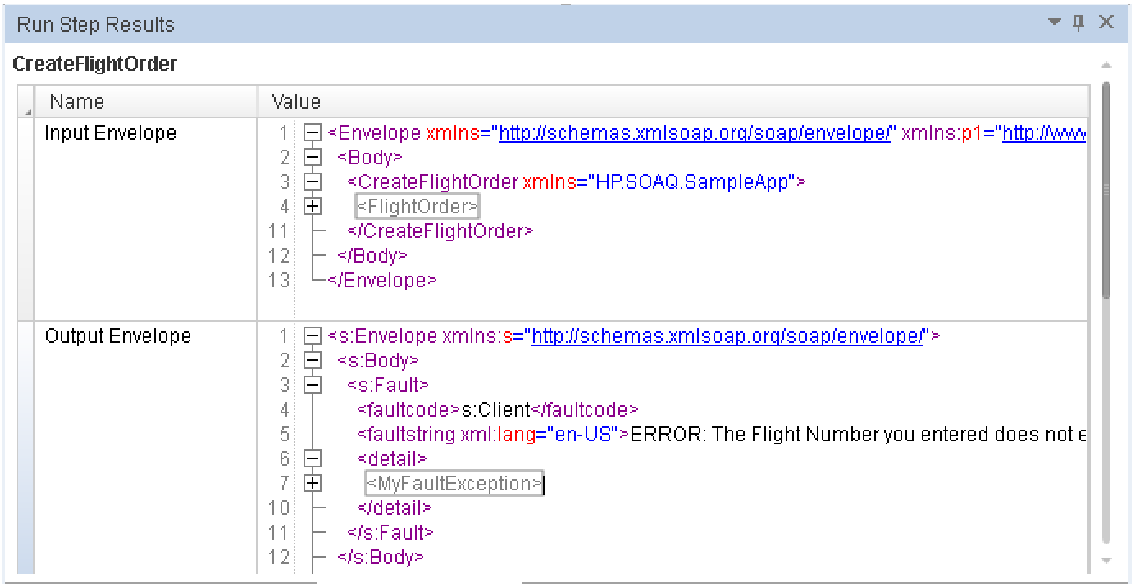


Concatenate Strings13

Name	Value
Name	Concatenate Strings13
Comment	
Prefix	Hello
Suffix	World
Result	Hello World
Invocation Result	Successfully concatenated strings

For services using SOAP, this pane shows the Input and Output envelopes. You can expand and collapse the nodes of the envelope to make it more readable.

This can be especially useful in checking the validity of your input property values. In the following example, the specified flight number was not valid.



CreateFlightOrder

Name: Input Envelope

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:p1="http://www
  <Body>
    <CreateFlightOrder xmlns="HP.SOAQ.SampleApp">
      <FlightOrder>
    </CreateFlightOrder>
  </Body>
</Envelope>
```

Name: Output Envelope

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <s:Fault>
      <faultcode>s:Client</faultcode>
      <faultstring xml:lang="en-US">ERROR: The Flight Number you entered does not e
      <detail>
        <MyFaultException>
      </detail>
    </s:Fault>
  </s:Body>
```

The Run Step feature also verifies checkpoints that are selected in the Checkpoints pane, including those of Fixed array type.

For a user interface description, see "Run Step Results Pane User Interface" on the next page.

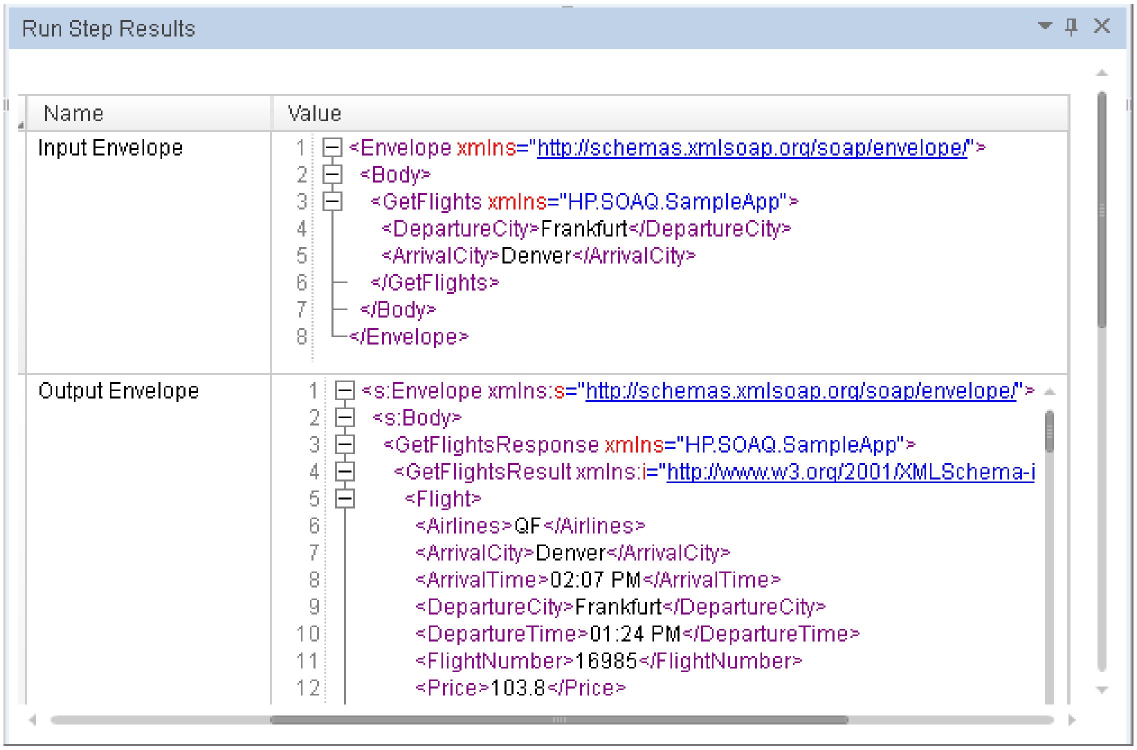
For REST methods, use the Run Method feature in the REST user interface. For details, see "Add/Edit REST Service Dialog Box" on page 1772.

Reference

Run Step Results Pane User Interface

Relevant for: API testing only

The Run Step Results pane shows the results of the run. For a simple built-in activity, it shows the status of the run. For SOAP messages, it shows the Input and Output envelopes.



Run Step Results

Name	Value

Input Envelope

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
 <Body>
  <GetFlights xmlns="HP.SOAQ.SampleApp">
   <DepartureCity>Frankfurt</DepartureCity>
   <ArrivalCity>Denver</ArrivalCity>
  </GetFlights>
 </Body>
</Envelope>
```

Output Envelope

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <GetFlightsResponse xmlns="HP.SOAQ.SampleApp">
   <GetFlightsResult xmlns:i="http://www.w3.org/2001/XMLSchema-i
    <Flight>
     <Airlines>QF</Airlines>
     <ArrivalCity>Denver</ArrivalCity>
     <ArrivalTime>02:07 PM</ArrivalTime>
     <DepartureCity>Frankfurt</DepartureCity>
     <DepartureTime>01:24 PM</DepartureTime>
     <FlightNumber>16985</FlightNumber>
     <Price>103.8</Price>
```

To access	Select **View > Run Step Results**
Relevant tasks	"How to Create an API Test" on page 1598
See also	• "The Canvas" on page 209 • "Run Step Dialog Box" on page 459

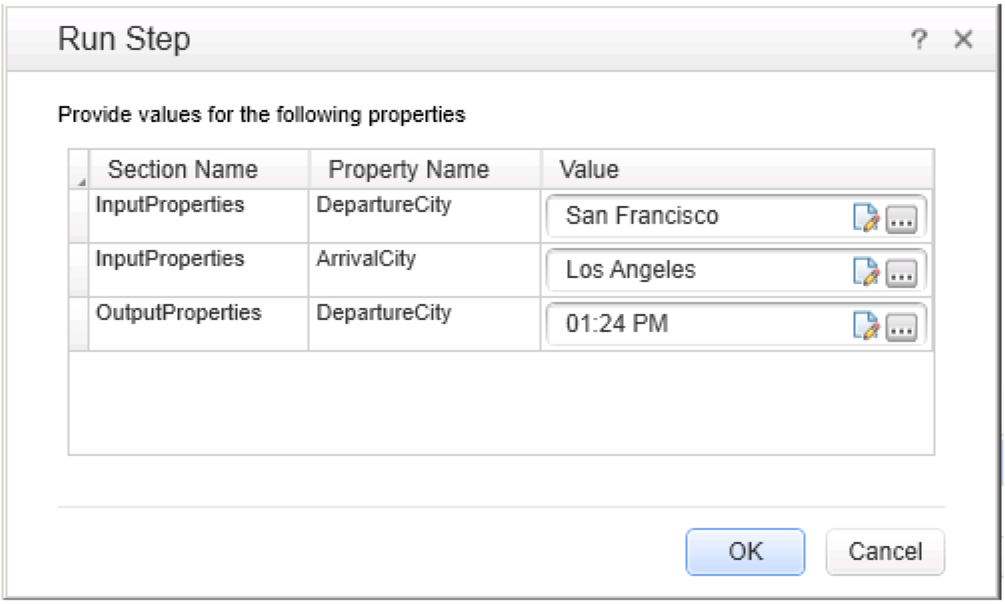
The user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Step Name> table	A table showing the names and values of properties and an invocation summary. The following is a list of the common fields: • **Name.** The name of the step as it appears in the General properties. • **Comment.** The text in the General properties tab's **Comment** field. • **<Properties>.** The input and output property names and values (non SOAP steps). • **Input Envelope.** A tree hierarchy of the SOAP envelope for the input properties. • **Output Envelope.** A tree hierarchy of the SOAP envelope for the output properties. • **Invocation Results.** Text describing the results of the run—both upon success and failure.
Checkpoints (<success ratio>) table	A table showing the checkpoints, with the following information: • **Property name** • **Expected** and **Actual** values • **Operator (=, <, >, and so forth)** • **XPath of the property**

Run Step Dialog Box

Relevant for: API testing only

This dialog box enables you to set constant values for the **Run Step** operation.



Run Step

Provide values for the following properties

Section Name	Property Name	Value
InputProperties	DepartureCity	San Francisco
InputProperties	ArrivalCity	Los Angeles
OutputProperties	DepartureCity	01:24 PM

To access	Do the following: 1. Add a step to the canvas that supports **Run Step.** 2. Link one or more properties to an output property from a previous step. 3. Click the step in the canvas and select **Run Step** from the right-click menu.
Important information	A property is listed when one of the following applies: - The property's input value is linked to the output property of a previous step. - The property requires a value, such as a URL for an HTTP Request step, and that value was not specified in the Properties pane. **Note:** When you assign constant values to the properties, the value is only used for this local **Run Step** step—the link to the output property in the Properties pane is not affected.
Relevant tasks	"How to Create an API Test" on page 1598.
See also	"Run Step Results Pane" on page 454

User interface elements are described below:

UI Element (A-Z)	Description
📝	**Open Text Editor.** Opens an editor for editing text strings. **Note:** For properties with enumeration, this button is not shown. Instead, a drop down list is available for selecting a value.
...	**Import from File.** Allows you to import a property value from a file. This is useful for complex properties whose values are stored in XML files. **Note:** For properties with enumeration, this button is not shown. Instead, a drop down list is available for selecting a value.
<property list>	A list of the properties to which a data source was assigned, that need to be given a constant value. It shows the following information: • **Section Name.** The origin or type of property, usually Input or Output. • **Property Name.** The property name as it appears in the Properties pane. • **Value.** The value to use in the run. By default, it takes the value from the first row of the data source.

Troubleshooting and Limitations - Running Steps and Run Step Results

Relevant for: API testing only

This section describes troubleshooting and limitations for the Run Step Results pane.

- When working in non-English operating systems, certain entries in the Run Step Results pane are hard-coded in English and not translated.
- The Run Step command is not supported for Java or JMS activities.

Chapter 15: Search Results Pane

Relevant for: GUI testing and API testing

This chapter includes:

Concepts

Search Results Pane Overview

Relevant for: GUI tests and components and API testing

The Search Results pane displays all occurrences of the search criteria you define using the Find dialog box or other Search menu items. Search results are displayed in one continuous list or grouped according to source file.

You can browse through the search results, locate specific results in their source files, and perform recent searches again to retrieve updated results.

For details on searching for specific strings, see "How to Navigate Through the Search Results Pane" on the next page.

For user interface details, see "Search Results Pane User Interface" on page 466.

Note: Searches are not supported in the Keyword View or the canvas.

Tasks

How to Navigate Through the Search Results Pane

Relevant for: GUI tests and components and API testing

This task describes how to navigate through results displayed in the Search Results pane, as well as how to access the recent search history.

This task includes the following steps:

- "Prerequisite" below
- "Modify the search results display" below
- "Browse and locate search results" on the next page
- "Display updated results for recent searches" on the next page
- "Clear the recent search history" on the next page

For user interface details, see "Search Results Pane User Interface" on page 466.

Prerequisite

Do one of the following:

- Perform a search using the **Find All** button in the "Find Dialog Box (Document Pane Editor) " (described on page 342). For details, see "How to Find or Replace Strings in Files" on page 327.
- **API testing only**: Search for references using the Search menu options. For details, see "How to Search for References or Classes in Documents in the Editor (API Testing Only)" on page 325.

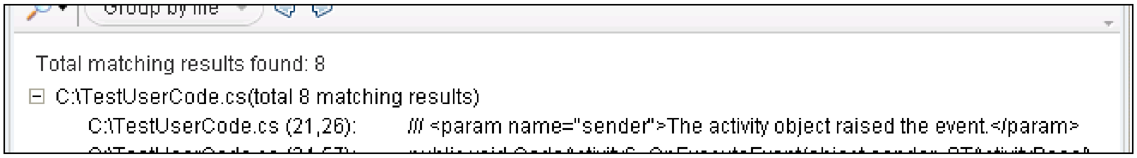
Modify the search results display

To toggle between display modes:

Click the **Select search list mode** button to toggle between the following display modes:

- **Flat list.** Displays all search results in a single list.
- **Grouped per file.** Displays all search results in a hierarchical list according to their source files.

When the results are grouped per file, a file information line is displayed above each group of results, indicating the source file and number of occurrences found in the file. For example:



```
Total matching results found: 8
⊟ C:\TestUserCode.cs(total 8 matching results)
      C:\TestUserCode.cs (21,26):      /// <param name="sender">The activity object raised the event.</param>
```

To collapse or expand the results per file:

Click the **Collapse per file** or **Expand per file** ⊞ ⊟ icon to the left of the file information line to view the results found in a specific file.

Browse and locate search results

- Use the scroll bar to move up or down in the search results list.
- Do one of the following to jump to the location of a search result in the document pane and highlight the search string:
  - Double-click a search result in the list.
  - Click **Previous** to select the previous item in the search results list.
  - Click **Next** to select the next item in the search results list.

If the file containing the search result is closed, it is automatically opened in the document pane. If you switch to the Keyword View, the entire row containing the search result is highlighted.

Display updated results for recent searches

Click the **Show last searches** button and select a search string from the drop-down list.

The **Show last searches** drop-down list displays the last ten search strings used.

Clear the recent search history

Click the **Show last searches** button and select **Clear history.**

The **Show last searches** drop-down list is cleared of all recent search strings.

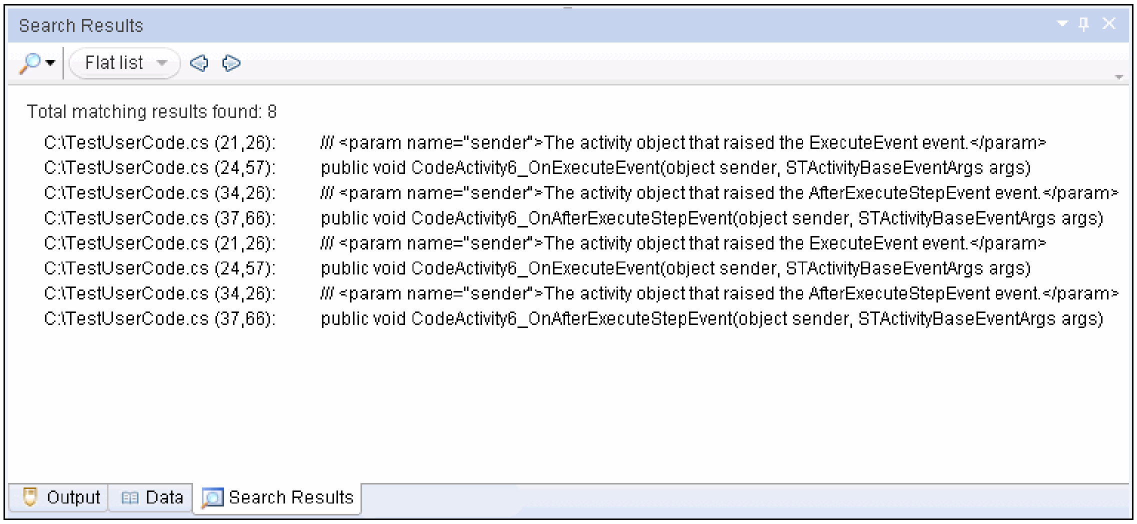
Reference

Search Results Pane User Interface

Relevant for: GUI tests and components and API testing

This pane displays all occurrences of the search criteria you defined and enables you to:

- Locate specific search results in their source files
- Retrieve updated search results for recent searches
- Clear the search history



```
Total matching results found: 8
  C:\TestUserCode.cs (21,26):     /// <param name="sender">The activity object that raised the ExecuteEvent event.</param>
  C:\TestUserCode.cs (24,57):     public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs args)
  C:\TestUserCode.cs (34,26):     /// <param name="sender">The activity object that raised the AfterExecuteStepEvent event.</param>
  C:\TestUserCode.cs (37,66):     public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseEventArgs args)
  C:\TestUserCode.cs (21,26):     /// <param name="sender">The activity object that raised the ExecuteEvent event.</param>
  C:\TestUserCode.cs (24,57):     public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs args)
  C:\TestUserCode.cs (34,26):     /// <param name="sender">The activity object that raised the AfterExecuteStepEvent event.</param>
  C:\TestUserCode.cs (37,66):     public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseEventArgs args)
```

To access	Do one of the following**:** • Select **View > Search Results** to view the results of your last search. • In the **Find** dialog box, define the search criteria and click **Find All**. • **API testingonly:** Perform a search for references using the Search menu options.
Important information	Click on a row or use the **Previous** and **Next** buttons to jump to the location of the search result in the document pane and highlight the search string. **Note:** Searches are not supported in the Keyword View or in the canvas.

Relevant tasks	- "How to Navigate Through the Search Results Pane" on page 464 - "How to Find or Replace Strings in Files" on page 327 - "How to Search for References or Classes in Documents in the Editor (API Testing Only)" on page 325
See also	"File and Item Types Included in String Searches" on page 317

The following sections describe:

- "Main User Interface Elements" below
- "Context Menu User Interface Elements" on the next page

Main User Interface Elements

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

UI Element	Description
🔍▾	**<Show recent searches>.** Enables you to: - View a drop-down list displaying the last ten **Find All** search strings used. - Clear the search history entirely. Select a search string to retrieve the most recent results, or select **Clear search history** to clear the recent search history.
Flat list	**<Search list display mode>.** Enables you to display all search results in a single list or grouped according to source file.
◁	**Previous result.** Selects the previous item in the search results list, jumps to the location of the search result in the document pane, and highlights the search string.
▷	**Next result.** Selects the next item in the search results list, jumps to the location of the search result in the document pane, and highlights the search string.
◢	**<Collapse or expand per file>.** Enables you to collapse or expand the list of search results located in a specific file, when viewing an expanded list of results grouped by file. **Note:** This icon is displayed when the results are grouped by file.

UI Element	Description
<Search results list>	Search results are displayed with the following syntax, depending on the type of source file: **GUI actions and function libraries for both grouped by file and flat list views** <File Path> (<line>,<column>): <text> **Note:** For function library results found in a solution or tests, the file path is the name of the function library. API test flow • **Grouped per file:** <Activity StepID>/<Tab Name>/<Property Name> : <Property Value> • **Viewed in flat list:** ▪ <Test Name>/<Activity StepID>/<Tab Name>/<Property Name> : <Property Value> or ▪ <Test Name><(Action)>/<Activity StepID>/<Tab Name>/<Property Name> : <Property Value> API user source code • **Grouped per file:** <line>,<column> : <text> • **Viewed in flat list:** <File Path> (<line>,<column>): < text >

Context Menu User Interface Elements

The user interface elements described below are available when you right-click a search result in the Search Results pane:

UI Elements	Description
Copy	Copies the content of the selected search result to the clipboard.

UI Elements	Description
Locate	Jumps to the location of the selected search result in the document pane and highlights the search string. **Note:** You can also double-click the search result.

Chapter 16: Solution Explorer Pane

Relevant for: GUI tests and components and API testing

This chapter includes:

Concepts

Solutions in UFT Overview

Relevant for: GUI tests and components and API testing

A **solution** is a collection of testing documents and other resources, similar to a binder or notebook. You can use solutions to organize your testing documents to help you perform comprehensive application testing.

For example, suppose you want to test a Web application for a flight booking application. You can create a solution containing several tests or components that verify various aspects of your application, such as logging in, booking a reservation, verifying the connection between your application and database, and verifying the transfer of booking information from your application to an airline server.

In UFT, all testing documents must be part of a solution. Therefore, when you create a testing document, you assign it immediately to a solution. You can provide descriptive names for your solutions, or you can accept the default names provided by UFT.

UFT solutions can includeGUI and API (non-GUI) tests, business components, business process tests and flows, application areas, function libraries, or user-created code documents. When you add a testing document to a solution, it retains its unique associations to other testing documents and resource files and does not create links to the other testing documents in that solution. The relationship between solutions and testing documents goes in one direction. The solution contains references to its testing documents, but the testing documents do not contain a reference back to any solutions that contain it.

You can create solutions for own use, or you can share solutions with other users by saving them in an accessible location, such as a network drive. Although your testing documents can be stored in the file system or ALM, solutions are always stored somewhere in the file system.

You view and manage solutions in the Solution Explorer pane.

Solution Explorer Pane Overview

Relevant for: GUI tests and components and API testing

You use the **Solution Explorer** pane to view and manage the testing documents in a solution. The Solution Explorer is displayed as a tree in the Solution Explorer pane. This tree displays a separate node for each type of testing document, with sub-nodes for all associated or referenced items.

From the Solution Explorer, you can:

- Add new or existing testing documents to a solution.
- Open any testing document directly from the Solution Explorer. You can open several testing documents side-by-side in the same session.

- Display the properties and settings for the solution or any of its testing documents by highlighting the relevant node. This refreshes most of the panes in UFT as the Solution Explorer pane serves as the master pane in UFT.
- Add resource files to a GUI testing document.
- Prioritize GUI testing resource files.
- Add user-created functions to a test by adding a function library (GUI testing) or C# user code file (API testing).
- Add an external reference file toan APItest for a run session or add a Web reference for use in an API test (API testing).
- Rename the solution and many of its sub-nodes.
- Run a test or component from the solution.

> **Note:** When you open a testing document from outside of the Solution Explorer (from the File menu, for example), the testing document always opens in a new, untitled solution, even if it is part of the currently open solution. Therefore, if you want to open a testing document that is part of the current solution, and you do not want to close any other open testing documents, open the testing document from the Solution Explorer.

For an overview of solutions, see "Solutions in UFT Overview" on the previous page.

For task details, see "How to Manage Items in the Solution Explorer Pane" on the next page.

For user interface details, see "Solution Explorer Pane User Interface" on page 478.

Tasks

How to Manage Items in the Solution Explorer Pane

Relevant for: GUI tests and components and API testing

This task describes how to perform various operations in the Solution Explorer pane.

This task includes the following steps:

- "Prerequisite: Show the Solution Explorer pane" on the next page
- "Create a solution" on the next page
- "Open a solution" on the next page
- "Create a document" on the next page
- "Add a document to the current solution" on page 475
- "Open an existing document from the current solution" on page 475
- "Add a resource file to your test (GUI testing only)" on page 475
- "Open a resource file (GUI testing only)" on page 476
- "Add an object repository file to the current solution (GUI testing only)" on page 476
- "Remove the association of a resource file with the test or action (GUI testing only)" on page 476
- "Change the priority of a resource file (GUI testing only)" on page 476
- "Run a test or component from the current solution" on page 477
- "Display the GUI test properties (GUI testing only)" on page 477
- "View or hide the sub-nodes in the test" on page 477
- "Remove a document from the current solution" on page 477
- "Save an individual document within the solution" on page 477
- "Save the solution" on page 477
- "Close the solution" on page 477

Prerequisite: Show the Solution Explorer pane

Do one of the following:

- Select **View > Solution Explorer**.
- Click the **Solution Explorer** button in the toolbar.
- Click the **Solution Explorer** tab in the main UFT window.

Create a solution

Do one of the following:

- Select **File > New > Solution**.
- Click the **New** button down arrow in the toolbar and select **New Solution**.
- Select **File > New** and choose the type of document to create.

In the "New <Document> Dialog Box" (described on page 152), enter a solution name in the **Solution Name** field and select the **Create directory for solution** option.

A new solution, also containing the type of document selected, opens in the Solution Explorer pane.

> **Note:** If you do not create a named solution, a generic solution called Solution Untitled is created.

Open a solution

Select **File > Open > Solution**. In the dialog box, select the name of your solution file.

If a document included in a solution is unavailable, due to issues with ALM connectivity or resource file problems, it appears in gray and an error describing the problem is displayed in the Errors pane. Right-click **Reload** after resolving the error to add the document to the solution for editing.

> **Note:**
>
> - If you open a solution when another solution is open, the first solution is closed and you are prompted to save any unsaved changes to documents contained in the closed solution.
> - You can save and open solutions only on the file system.

Create a document

Right-click the **<solution name>** node and select **Add > Add New Test/Business Component/Function Library/Application Area**.

The "Add Test/Component to Solution Dialog Box"(described on page 146) opens, enabling you to create a new document and add it into your solution file.

Add a document to the current solution

Right-click the **<solution name>** node and select **Add > Add Existing Test/Business Component/Function Library/Application Area**.

The "Add <Existing Document> to Solution Dialog Box" (described on page 145) opens, enabling you to add a document to your solution file.

> **Note:** You can add a maximum of 10 documents to a solution.

Open an existing document from the current solution

Double-click a node for your document:

- **For GUI tests or components:** the **<test name>**/**<component name>** node.
- **For GUI actions:** the **<action name>** node located under the **<test name>** node.
- **For application areas:** the node containing the application area.

  Application area nodes are located in different places depending on their association with a component:

  - For an independent application area, click the **<application area name>** found in the Additional Items folder.
  - For an application area associated with a component, click the **<application area name>** node under the component.
- **ForAPI tests and components:** the **Flow** node located under the test or component node.
- **ForAPIevent handlers:** the **Events** node located under the test or component name.
- **ForAPI user code files:** the <**User Code file name>** node located under the test or component name.

A tab opens in the document pane with the selected document.

Add a resource file to your test (GUI testing only)

To add a resource file to your test, do one of the following:

- **For a function library:** Right-click the test node and select **Associate Function Library**.
- **For a recovery scenario:** Right-click the test node and select **Associate Recovery Scenario**.

The "Open/New <Document>/<Resource> Dialog Box" (described on page 156) opens. Browse to and select the resource file to add.

Open a resource file (GUI testing only)

Double-click the selected node for your resource file:

- **For function libraries:** the node containing the function library.

  The function library nodes are located in different places, depending on their association with a test, component, or application area:

  - For a function library associated with a test or application area, click the **<function library name>** found in the Function Libraries node in the test/application area node.
  - For a function library not associated with a test or application area, click the **<function library name>** found in the Solution Items folder node.
- **For an object repository:** the **<object repository name>** found under an action, component, or application area node.
- **For a recovery scenario:** the **<recovery scenario name>** found under the **Recovery Scenarios** node in an action or component node.

Add an object repository file to the current solution (GUI testing only)

Right-click on an **<action name>** node and select **Associate Repository with Action**.

In the "Open/New <Document>/<Resource> Dialog Box" (described on page 156), select the object repository to add to the current action.

Remove the association of a resource file with the test or action (GUI testing only)

Right-click the resource node and select the relevant command:

- **Remove Function Library from List**. Removes the association of the selected function library with your test.
- **Remove Recovery Scenario from List.** Removes the selected recovery scenario from your test.
- **Remove Repository from List**. Removes the selected object repository from the action.

Removing a resource node severs the association between the resource and the test or action. The resource node is deleted from the Solution Explorer pane and any other location that indicates associations, such as the Test Settings dialog box (for function libraries and recovery scenarios) and the Associate Repositories dialog box (for shared object repositories).

Change the priority of a resource file (GUI testing only)

Right-click the resource node and select **Move Up** or **Move Down**.

> **Note:** By default, resources are listed in priority order in the order in which they were associated with a test, component, or application area.

Run a test or component from the current solution

Right-click on a test or component node and select **Run**. The Run dialog opens, enabling you to choose preferences for your run session. For details, see "Run Dialog Box" on page 651.

Display the GUI test properties (GUI testing only)

Right-click the **GUITest** node in the tree and then select **Settings** to open the Test Settings dialog box. Details of the test and its path are displayed. For details, see "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.

View or hide the sub-nodes in the test

Do one of the following:

- Right-click the **Test** node in the tree and select **Expand All** or **Collapse All**.
- Click the expand or collapse icons next to the node to expand or collapse the node.

Remove a document from the current solution

Do any of the following:

- Right-click a document node and select **Remove from Solution**.
- Select a document and press DELETE.

> **Note:** Removing a document from a solution does not delete it from your file system or ALM project.

Save an individual document within the solution

Right-click a document name and select **Save As**.

The "Save <Resource>/Save <Document> As Dialog Box"(described on page 164) opens, enabling you to save the document with a different name or in a different location.

Save the solution

Select **File > Save All** or press the **Save All** button on the toolbar.

> **Note:** Selecting this command also saves any modified document within the solution.

Close the solution

Select **File > Close Solution**.

> **Note:** Closing a solution closes all documents that are part of the solution file. These files are not deleted from your file system or ALM project when a solution is closed.

Reference

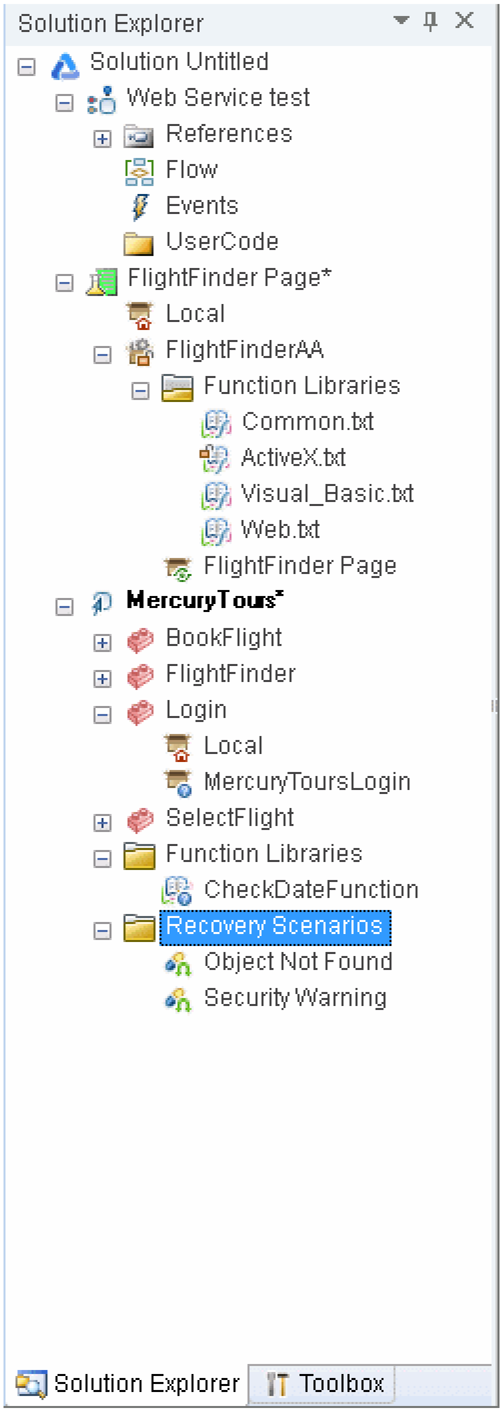
Solution Explorer Pane User Interface

Relevant for: GUI tests and components and API testing

This pane enables you to:

- View the currently open solution, and its documents, and resource files
- Add or remove items from a solutions
- Open documents contained in a solution
- Associate GUI resource files to a specific test or component
- Run a test or component

The following image shows the Solution Explorer pane displaying a node for the different types of GUI testing documents and their resource files.



To access	Do one of the following: • Select **View > Solution Explorer** • Click the **Solution Explorer** toolbar button 🗔.

Important information	- The Solution Explorer pane is displayed by default when you start UFT. - The Solution Explorer pane is the control pane within UFT. Other panes, menus, and toolbars update their content to match the document selected in the Solution Explorer pane. - Document nodes are displayed in alphabetical order. - The active document is shown as bold in the Solution Explorer pane. - **Note for GUI testing:** Resource nodes contained under a document node are displayed in alphabetical order, but specific resources files are displayed in the order in which they were associated to a document. You can use the context menu for each resource node to change the priority order of resource files.
Relevant tasks	"How to Manage Items in the Solution Explorer Pane" on page 473
See also	"Solution Explorer Pane Overview" on page 471

The Solution Explorer pane can display the following key nodes:

Note: All user interface descriptions of each node also contain the descriptions of the context menus available for the node.

Solution Node (Solution Explorer Pane)

Relevant for: GUI tests and components and API testing

The **solution node** displays the name of the solution and contains all the tests, component, application area, and resource nodes necessary to edit and run a test or component. For details about a specific type of document node, see the relevant section on each node below.

The following context menu options are available when you right-click the solution:

- **Add New/Existing <Document>.** Opens the "Add Test/Component to Solution Dialog Box" (described on page 146) or "Add <Existing Document> to Solution Dialog Box" (described on page 145), which adds a new or existing document to a solution.

> **Note:** You can add a maximum of 10 documents to a solution.

- **Expand All.** Expands all test, component, or application area sub-nodes contained in the solution.
- **Collapse All.** Collapses all sub-nodes contained in the solution.
- **Compile solution (API testing only)** Compiles the solution files that have changed.
- **Recompile solution (API testing only).** Compiles all solution files, regardless of whether they changed. You should run the **Clean solution** command before this command.
- **Clean solution (API testing only).** Removes all files generated in the previous compilation of the solution.

> **Note:**
>
> - The **Compile**, **Recompile**, and **Clean** commands are applicable only if your solution contains API UFT tests. This command is not available if a solution contains only GUI tests or components.
> - Compiling GUI tests is not supported.

API Test/Component Node (Solution Explorer Pane)

Relevant for: API testing only

This **API Test node** is available only if your solution contains an API test or component. Some of the nodes contained within this one, are available only if a relevant resource is created in a test or component.

This node can contain the following sub-nodes:

- "References Node" on the next page
- "Flow Node" on the next page
- "Events Node" on the next page
- "Action Node" on the next page
- "External Tests/Actions Node" on the next page

- "UserCode Node" on the next page
- "Context Menu Options" on the next page

References Node

This node contains all of the .dll files used by UFT to runAPI tests.

> **Note:** Do not remove any of the reference files contained by default in this node. They are required to perform anAPI run session. You can add additional user-created reference files using the context menu options. For details, see "Add Reference Dialog Box" on page 485.

Flow Node

This node provides a link to the API test canvas, which displays the test's flow. Double-clicking this node displays the canvas or opens the canvas's tab (if closed). Each individual action also contains a flow node for the action's steps.

For details on using the canvas, see "The Canvas" on page 209.

Events Node

This node provides a link to the test or action's TestUserCode.cs file used for coding event handlers. Double-click on this node to display the file. Each individual action within a test also has an events node for the action.

> **Note:** If you double-click on this link independently of the events tab in the Properties pane, the global test or component user code is displayed, including all event handlers currently written into the test.

For details on coding API testing events, see "Writing Code for API Test Events - Overview" on page 1938.

Action Node

This node contains all of the actions created in anAPI test.

Each action node contains a **Flow** and **Events** node.

> **Note:** This node is displayed only when you create an action in anAPI test. For details, see "Actions in API Testing - Overview" on page 1787.

External Tests/Actions Node

This node contains all of the tests or actions (both GUI and API) called from the API test. The different types of called actions are noted by their icons. For details on the different GUI action icons, see "The Canvas" on page 209.

> **Note:** This node is only displayed when you create a call to an external test or action. For details on using HP Functional Testing activities in an API test, see "HP Automated Testing

Tools Activities" on page 1714.

UserCode Node

This node contains any user code files associated with the test. The following user interface elements are available:

UI Element	Description
<file name>	The name of the user code file name. **Note:** Only .cs files can be associated with anAPI.
<folder name>	The name of a folder containing user-generated code to add to a test.

Context Menu Options

The following context menu options are available for the **API Test** node:

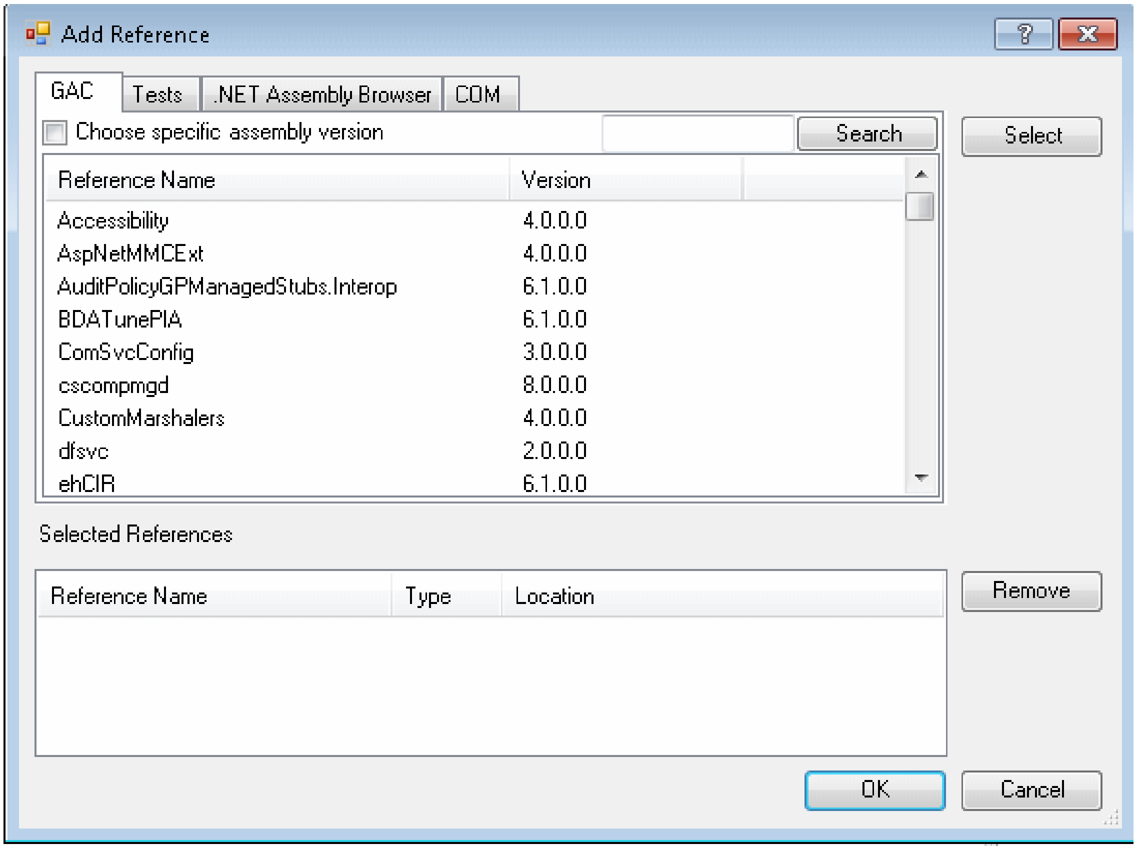
Node	Context Menu Option
<test name>	• **Run.** Opens the "Run Dialog Box" (described on page 651) to begin a run session for the current test. • **Expand All.** Expands all sub-nodes found under the test. • **Collapse All.** Collapses all sub-nodes found under the test. • **Remove from Solution.** Removes the current test from the solution file. • **Save As.** Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164), enabling you to save the selected test under a different name or in a different location.
References folder	• **Add Reference.** Opens the "Add Reference Dialog Box" (described on page 485), which adds a external reference file to your test.
<reference name>	• **Refresh.** Updates the selected reference file with any changes made to the file. • **Remove.** Removes the current reference from the list. • **Properties**. Displays the properties of the reference file in the Properties pane. **Caution:** Do **not** remove the reference files included in this node, as doing so may cause a run session to fail.

Node	Context Menu Option
<action name>	- **Delete Action.** Removes the selected action from the test. - **Rename Action.** Opens the "Rename Action Dialog Box" on page 1798, enabling you to rename the action. **Note:** Before renaming an action, you must close the action's tab and user code file and save your test.
User Code	- **Add New File.** Opens a dialog box to add a new custom code file to your test. For user interface details, see "Open/New <Document>/<Resource> Dialog Box" on page 156. - **Add Existing File.** Opens the "Add <Existing Document> to Solution Dialog Box" (described on page 145), which adds an existing custom code file to the test.
<user code file/folder name>	- **Open.** Opens the file in the document pane or brings it into focus. - **Open Containing Folder in Explorer.** Opens the folder containing the user code file in Windows Explorer. - **Remove.** Removes the user code file from the user code node and deletes it. - **Exclude From Test.** Removes the association of this user code file from your test. - **Rename.** Renames the selected file. - **Properties.** Opens the file's properties in the Properties pane.

Add Reference Dialog Box

Relevant for: API testing only

This dialog box enables you to add a reference file to your API test.



Reference Name	Version
Accessibility	4.0.0.0
AspNetMMCExt	4.0.0.0
AuditPolicyGPManagedStubs.Interop	6.1.0.0
BDATunePIA	6.1.0.0
ComSvcConfig	3.0.0.0
cscompmgd	8.0.0.0
CustomMarshalers	4.0.0.0
dfsvc	2.0.0.0
ehCIR	6.1.0.0

Reference Name	Type	Location

To access	Do one of the following: • Select the **References** node of an API test, right-click and select **Add Reference**. • Select **Design > Add Reference.**

User interface elements are described below:

UI Element	Description
GAC tab	Displays a list of the GAC (Global Assembly Cache) assemblies and version numbers. **Note:** • If you select the **Choose specific assembly version** option, the pane displays a list of all versions of all assemblies. You can then select a specific version of an assembly to insert in the test. • If you enter a text string into the **Search** field, UFT searches through the list of assemblies for the assembly that matches your search string.
Tests tab	A list of all tests contained in the current solution.
.NET Assembly Browser tab	Enables you to searches the file system for .NET assemblies to import into a test. The browser displays the following information: • **<directory location>.** The location on the file system for a folder or assembly. • **Back/Up** buttons. Enable you to return to the previously selected directory or the previous level on the directory hierarchy to search for an assembly. • **<files list>.** The list of all available files in a directory location.
COM tab	A list of all software components and their location in the file system. **Note:** If a path to a component is not listed in the pane, then the component is located in the <Unified Functional Testinginstallation>/bin folder.
Select	Adds the current assembly, project, or software component to the **Selected References** list.
Remove	Removes the current assembly, project, or software component from the **Selected References** list.
Selected References	A list of all assemblies, projects, or software components to add to your test. The grid displays the following information: • **Reference Name.** The reference name as displayed in the References node in the Solution Explorer pane. • **Type.** The type of reference for the selected reference, project, or assembly. • **Location.** The location of the selected reference in the file system. If a location is not displayed for the reference, it is located in the< **Unified Functional Testing installation>/bin** folder.

Business Process Test/Flow Node (Solution Explorer Pane)

Relevant for: business process tests and flows

The Solution Explorer pane does not display sub-nodes for business process tests or flows. You can view the components in the document pane.

The following context menu options are available for a business process test or flow:

- **Run.** Runs the current business process test or flow.
- **Remove from Solution.** Removes the current business process test or flow from the solution.

You can maintain your business process tests and flows in both UFT and ALM. For details, see "Business Process Testing in UFT - Overview" on page 2065 and the *HP Business Process Testing User Guide*.

GUI Test Node (Solution Explorer Pane)

Relevant for: GUI tests only

The **GUI test node** is available only if your solution contains a GUI test. Some of the nodes contained within this one, are available only if a relevant resource is associated with the test.

This node can contain the following sub-nodes:

- "Action Node" below
- "External Actions Node" below
- "External Tests Node" on the next page
- "Function Libraries Node" on the next page
- "Recovery Scenarios Node" on the next page
- "Context Menu Options" on page 489

Action Node

This node contains all of the actions in the GUI test.

Each action node contains a **Local** node, for its local object repository, and an additional node for each object repository associated with the action.

The order in which the object repositories are listed determines the priority of the object repository, the order in which UFT searches the object repositories for an object in a step.

External Actions Node

This node contains all of the external actions called by the GUI test.

External Tests Node

This node contains all of the external tests called by the GUI test.

Function Libraries Node

This node contains all of the function libraries associated with the test.

The order of the function libraries in the list determines the order in which UFT searches for a function or subroutine that is called from a step in your test or component.

Recovery Scenarios Node

This node contains all of the recovery scenarios associated with the test.

The order of the recovery scenarios in the list determines the order in which UFT searches for a relevant scenario when an error occurs during a run session.

Context Menu Options

The following context menu options are available for GUI Test node:

Node	Context Menu Option
Test	• **Add.** Adds action calls or associates resource files with a test. The following commands are available: ▪ **Add Call to New Action.** Opens the "Insert Call to New Action Dialog Box (GUI Testing)""Insert Call to New Action Dialog Box (GUI Testing) "described on page 913). ▪ **Add Call to Copy of Action/Add Call to Existing Action.** Opens the "Select Action Dialog Box " (described on page 915). ▪ **Associate Function Library/Associate Recovery Scenario.** Opens the "Open/New <Document>/<Resource> Dialog Box" (described on page 156). • **Run.** Opens the "Run Dialog Box" (described on page 651). • **Expand All.** Expands all sub-nodes under the **<test name>** node. • **Collapse All.** Collapses all sub-nodes under the **<test name>** node. • **Remove from Solution.** Removes the current test from the solution. **Note:** Removing a test from a solution does not delete it from your file system or ALM project. • **Save As.** Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164), which enables you to save a test with a different name or in a different location. • **Settings.** Opens the "Properties Pane (Test/Business Component Settings Dialog Box)""Properties Pane (Test/Business Component Settings Dialog Box)"described on page 590).

Node	Context Menu Option
Action	**Note:** Many of these options are applicable to external action nodes also. - **Associate Repository with Action.** Opens the "Open/New <Document>/<Resource> Dialog Box" (described on page 156), which adds the object repository to the selected action. - **Rename.** Renames the selected action. - **Delete Action.** Deletes the action from the current test. - **Properties.** Opens the action's properties in the "General Properties Tab (Properties Pane - GUI Testing)" of the Properties pane (described on page 390).
Object Repository	**Note:** These options are applicable for object repositories that are associated with an action. - **Open Repository.** Opens the selected object repository in the "Object Repository Window" (described on page 1274). - **Remove Repository from List.** Removes the selected object repository from the list of associated object repositories. - **Move Up.** Moves the selected object repository up the priority list of associated repositories. - **Move Down.** Moves the selected object repository down the priority list of associated repositories.
<external tests list>	- **Add to current solution.** Adds the selected test to the solution and opens it in the document pane. - **Remove.** Removes the selected test from the calling test.
Function Libraries	**Associate Function Library.** Opens the "Open/New <Document>/<Resource> Dialog Box"(described on page 156) to associate an existing function library with the current test.

Node	Context Menu Option
<function libraries list>	• **Open Function Library.** Opens the selected function library or brings the document into focus in the document pane. • **Remove Function Library from List.** Removes the current function library from the associated function libraries list. • **Move Up.** Moves the current function library up the priority list of associated function libraries. • **Move Down.** Moves the current function library down the priority list of associated function libraries. • **Properties.** Opens the function library's properties in the "General Properties Tab (Properties Pane - GUI Testing)" of the Properties pane (described on page 390).
Associated Recovery Scenarios	**Associate Recovery Scenario.** Opens the "Add Recovery Scenario Dialog Box"(described on page 1120), creates or opens recovery scenarios for your test.
<recovery scenario list>	• **Recovery Scenario Properties.** Opens the "Recovery Scenario Properties Dialog Box"(described on page 1124), which displays details about the current recovery scenario. • **Remove Recovery Scenario from List.** Removes the current recovery scenario from the list of associated recovery scenarios. • **Move Up.** Moves the current recovery scenario up the priority list of associated recovery scenarios. • **Move Down.** Moves the current recovery scenario down the priority list of associated recovery scenarios. • **Enable/Disable Recovery Scenario.** Enables or disables the current recovery scenario. Each recovery scenario displays an icon to indicate whether it is enabled or disabled.

GUI Component Node (Solution Explorer Pane)

Relevant for: GUI components only

The **GUI component node** is displayed only if your solution contains a keyword GUI component or a scripted GUI component. Some of the nodes contained within this one, are available only if a relevant resource is associated with the test.

This node can contain the following sub-nodes:

- Local Repository node
- Associated Application Area node

This node lists all of the application areas associated with the component. Each application area node contains the following:

Option	Description
<application area name>	The name of the application area
Function Libraries node	A list of all function libraries associated with the application area of a component. By default, UFT displays the Common.txt, ActiveX.txt, Visual_Basic.txt, and Web.txt function libraries.
<associated shared repositories>	A list of all shared repositories associated with the application area of a component.

The following context menu options are available for a GUI component node:

Node	Context Menu Option
<component name>	- **Run.** Opens the "Run Dialog Box" (described on page 651), enabling you to run your component. - **Expand All.** Expands all sub-nodes contained under the **<component name>** node. - **Collapse All.** Collapses all sub-nodes contained under the **<component name>** node. - **Remove from Solution.** Removes the current component from the solution. - **Save As.** Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164), enabling you to save the component with a different name or in a different location. - **Change Application Area.** Opens the "Open/New <Document>/<Resource> Dialog Box" (described on page 156), enabling you to change the application area associated with the selected component. - **Settings.** Displays the component properties in the "General Properties Tab (Properties Pane - GUI Testing)" of the Properties pane (described on page 390).

Node	Context Menu Option
Object repository	• **Open Repository.** Opens the "Object Repository Window" (described on page 1274). • **Remove Repository from List.** Removes the local object repository from the list of associated repositories. • **Move Up.** Moves the selected object repository up the priority list of local object repositories. • **Move Down.** Moves the selected object repository down the priority list of local object repositories.
<application area>	• **Expand All.** Expands all sub-nodes contained under the <application area> node. • **Collapse All.** Collapses all sub-nodes contained under the <application area> node. • **Open Associated Function Library.**Opens the selected application area to the Function Libraries pane in order to enable you to open and edit any associated function libraries. • **Open Associated Object Repository.** Opens the selected application area to the Object Repositories pane in order to enable you to open and edit any associated object repositories. • **Add Application Area to Solution.** Adds the selected application area to a solution to enable you to edit it.
<associated function library name>	• **Open Function Library.** Opens the selected function library or brings it into focus in the document pane. • **Remove Function Library from List.** Removes the selected function library from the list of associated function libraries. • **Move Up.** Moves the selected function library up the priority list of associated function libraries. • **Move Down.** Moves the selected function library down the priority list of associated function libraries. • **Properties.** Opens the function library properties in the "General Properties Tab (Properties Pane - GUI Testing)" of the Properties pane (described on page 390).

Application Area Node (Solution Explorer Pane)

Relevant for: GUI components only

The **application area node** is displayed under the **Additional Items** folder. It is displayed only if your solution contains a separate application area independent of a component. The following user interface elements are displayed:

UI Element	Description
Function libraries node	A list of all function libraries associated with the application area of a component. By default, UFT displays the Common.txt, ActiveX.txt, Visual_Basic.txt, and Web.txt function libraries.
Shared Object Repositories	A list of all shared repositories associated with the application area of a component.

Note: Associating a resource file is done in the application area in the document pane. For details see "How to Create and Manage Application Areas" on page 2097.

The following context menu options are available:

Node	Context Menu Option
<application area name>	- **Expand All.** Expands all sub-nodes contained under the **<application area>** node. - **Collapse All.** Collapses all sub-nodes contained under the **<application area>** node. - **Associate Function Library.** Opens the "Function Libraries Pane (Application Area)" (described on page 2102). - **Associate Object Repository.** Opens the "Object Repositories Pane (Application Area)" (described on page 2105). - **Remove from Solution.** Removes the current application area from the solution. - **Save As.** Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164).

Node	Context Menu Option
<function library name>	• **Open Function Library.** Opens the selected function library or brings it into focus in the document pane. • **Remove Function Library from List.** Removes the selected function library from the list of associated function libraries. • **Move Up.** Moves the selected function library up the priority list of associated function libraries. • **Move Down.** Moves the selected function library down the priority list of associated function libraries. • **Properties.** Opens the function library's properties in the "General Properties Tab (Properties Pane - GUI Testing)" of the Properties pane (described on page 390).
<shared object repository>	• **Open Repository.** Opens the object repository in the "Object Repository Window" (described on page 1274). • **Remove Repository from List.** Removes the selected object repository from the application area. • **Move Up.** Moves the selected object repository up the priority list of associated shared object repositories. • **Move Down.** Moves the selected object repository down the priority list of associated shared object repositories.

Troubleshooting and Limitations - Solution Explorer Pane

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for the Solution Explorer pane.

Solutions stored on a network location are not locked when opened by UFT.

Chapter 17: Tasks Pane

Relevant for: GUI tests and components and API testing

This chapter includes:

Concepts

Tasks Pane Overview

Relevant for: GUI tests and components and API testing

The Tasks pane enables you to create and manage TODO comments for issues that need to be handled in yourGUI actions, GUI components, function libraries, and user code files.

TODO comments are reminders that are inserted as comments adjacent to the relevant steps in your testing document. For example, you can provide instructions to someone else during a handover, or you can remind yourself to do something.

When you create event handlers in an API test, the editor automatically inserts TODO text, indicating where you need to enter your code.

You can access TODO comments from the Tasks pane or directly from the testing document. For details, see "Tasks Pane User Interface" on page 501.

If needed, you can export your TODO comments to an XML file or Microsoft Excel.

Tasks

How to Create and Manage TODO Comments

Relevant for: GUI tests and components and user code files

This task describes the different operations you can perform to manage TODO comments inGUI actions, GUI components, function libraries, or user code files.

- To **add** a new TODO comment, insert a comment adjacent to the relevant step in your document. A comment can begin with any of the following permutations of the words **to do**: To Do, todo, to-do, or TODO.

  For details on comments in GUI testing, see "Comments in the Keyword View" on page 922 and "Comments in VBScript" on page 1017.

- To **delete** a TODO comment, you must delete the source line in the document. This removes the TODO comment from the pane. (To jump to the source line, you can either double-click a comment in the Tasks pane, or you can highlight the comment in the Tasks pane and then click the **Locate** button.)

- To **sort** by a specific column, click on the column header.

- To **rearrange columns**, drag a column header to a different location.

- To **filter** TODO comments, click the **Show Tasks from** button and select one of the following:

  - **Solution.** Displays all of the TODO comments stored in all of the tests included in the solution (including closed tests).

  - **A specific test or component.** Displays only the TODO comments that are stored in that test or component.

  - **Function Libraries.** Displays only the TODO comments that are stored in any function libraries that are currently open and any function libraries associated to tests in the solution.

- When showing TODO comments from a solution or a specific GUI test, you can **show or hide** comments stored in external actions or associated function libraries, by clicking the and/or toggle buttons, respectively. For more details, see "Tasks Pane User Interface" on page 501.

- To **export** TODO comments:

  a. Click the **Export Task List** button.

  b. In the "Save <Resource>/Save <Document> As Dialog Box" on page 164 (described on page 164), browse to the required location in the file system.

c. Enter a file name and specify the file type. You can export the file as XML, CSV, or .xls/.xlsx (if Excel is installed on the computer).

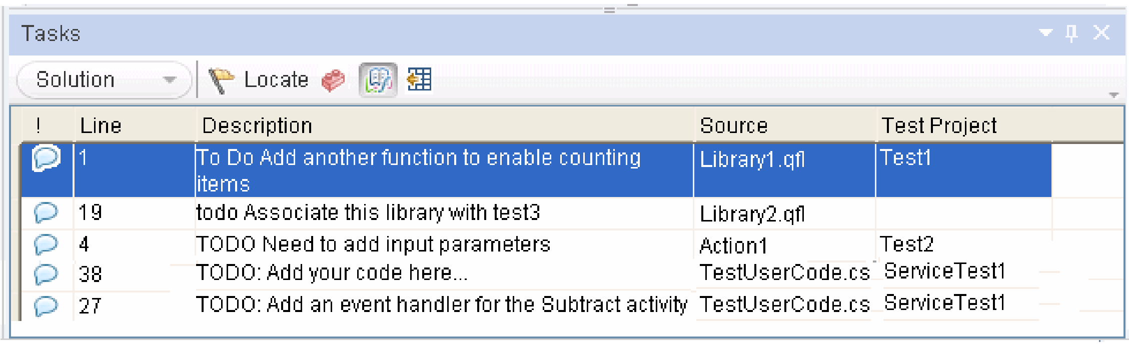
d. Click **Save**. The TODO comments are saved to a file in the specified location and in the specified format.

Reference

Tasks Pane User Interface

Relevant for: GUI tests and components and user code files

This pane enables you to view and access TODO comments inGUI actions, GUI components, function libraries, or user code files.



!	Line	Description	Source	Test Project
	1	To Do Add another function to enable counting items	Library1.qfl	Test1
	19	todo Associate this library with test3	Library2.qfl	
	4	TODO Need to add input parameters	Action1	Test2
	38	TODO: Add your code here...	TestUserCode.cs	ServiceTest1
	27	TODO: Add an event handler for the Subtract activity	TestUserCode.cs	ServiceTest1

To access	In the main UFT window, select **View > Tasks**.

Important information	- This view can display any comment step that begins with any of the following permutations of the words **to do**: To Do, todo, to-do, or TODO (not case-sensitive). **Example:** To Do need to ask Sarah to add design steps - The text displayed in the Comments view is limited to 260 characters. If the text exceeds this limit, and you want to view the entire comment, you can jump to the comment in the testing document by double-clicking the comment line in the Comments view. ▪ Local GUI actions and GUI components contained in the solution or in the selected GUI test. ▪ Any currently open GUI component. ▪ Any currently open function library. ▪ Any function library associated with a selected GUI test or component, or with any GUI test or component in the solution. ▪ Any external actions associated with a selected test or with any test in the solution. ▪ Any user code file contained in the current solution or in the selected API test or component. - By default, this view displays all TODO comments in any GUI actions, GUI components, and user code files contained in the solution.
Relevant tasks	"How to Create and Manage TODO Comments" on page 499
See also	"Tasks Pane Overview" on page 498

User interface elements are described below:

Toolbar Buttons

	Toolbar Option	Description
	<Filter>	Enables you to show all of the TODO comments for the solution, or to filter the display to show TODO comments only for a specific test or component, or only for function libraries.

	Toolbar Option	Description
	Locate	Jumps to the comment line in theGUI action, GUI component, function library, or user code file for the currently selected TODO comment. **Tip:** You can also double-click a TODO comment to jump to its location in the source document. **Note:** This option is available only when a line in the Tasks pane is selected.
	Show Comments in External Actions	Toggle button that enables you to display or hide any TODO comments from external actions, when showing TODO comments from a solution or test.
	Show Comments in Function Libraries (GUI testing only)	Toggle button that enables you to display or hide any TODO comments from function libraries (in addition to other TODO comments displayed in the pane).
	Export Task List	Saves the TODO comments to an external file, such as a text file. You can save the list of TODO comments in any of the following formats: • XML (Extensible Markup Language) • XLS/XLSX (Microsoft Excel file) • CSV (Comma-Separated Values file)

Columns

Column	Description
Line	The line number of the TODO comment in the source document.
Description	The text of the TODO comment.
Source	The name of the GUI action or component, or the file name of the function library or user code file containing the TODO comment.
Test	The name of the test containing the TODO comment, or associated with it.

Context Menu Options

Context Menu Option	Description
Copy	Copies the TODO comment to the Clipboard.
Locate	Jumps to the comment line in theGUI action, GUI component, function library, or user code file for the currently selected TODO comment. **Tip:** You can also double-click a TODO comment to jump to its location in the source document.

Chapter 18: Toolbox Pane

Relevant for: GUI testing, API testing, and business process testing

This chapter includes:

Concepts

Toolbox Pane Overview

Relevant for: tests, components, actions, function libraries, and flows

The **Toolbox** pane contains items that you can use to create steps in your testing document.

The items available in the toolbox depend on the document that is currently in focus in the document pane.

For GUI testing: The Toolbox pane displays the test objects and functions available to the current action, component, or function library. When a GUI test is in focus, the toolbox is empty.

You can view these items, open them in the object repository, action, or function library in which they are defined, or use them to create steps in your document.

Drag and drop, or copy and paste, a test object or function into the document pane to add a step your GUI testing document.

- When you drag and drop a test object into an action or component, a step is inserted with the default operation for that test object.
- When you drag and drop a function into an action, component, or function library, a call to that function is inserted.

Examples of usage

- If you drag and drop a button object into an action or component, a step is added using the button with a **Click** operation (the default operation for a button object).
- If you drag and drop the function count_items (obj, num) into an action, component, or function library, a comment and a call to that function are added:

```
' count_items (obj, num)
count_items
```

For API Testing: The Toolbox pane provides a collection of standard service activities for functional testing in areas such as XML, SOAP, Java, JMS, IBM MQWebsphere and HTTP. You can add more activities to the Toolbox pane by importing WSDLs or using services from the repository.

After you import a Web service, it appears in the Toolbox pane, under the **Local Activities > Web Services** node. When you expand the Web Services node, the Toolbox pane displays the service's name, port, and operations.

You can also create new custom activities, using the built-in Activity Wizard. For details, see "API Testing Extensibility" on page 2022.

Drag and drop an activity from the Toolbox pane into the document pane to add a step to your API test or action.

For business process testing: The Toolbox pane displays the components and flows that are available for you to add to the current business process test or flow. Double-click or drag and drop a component or flow to add it to BPT test or flow open in the document pane.

Toolbox items are listed according to their ALM path, and you can use the search bar to find a specific component or flow by name.

Tasks

For details about the API testing Toolbox pane, see "Toolbox Pane User Interface (API Testing) " on page 514.

For details about the business process testing Toolbox pane, see "Toolbox Pane User Interface (BPT in UFT)" on page 518.

How to Work with the Toolbox Pane (GUI Testing)

Relevant for: GUI actions, components, and function libraries

For details about the API testing Toolbox pane, see "Toolbox Pane User Interface (API Testing) " on page 514.

For details about the business process testing Toolbox pane, see "Toolbox Pane User Interface (BPT in UFT)" on page 518.

This task describes how to work with the Toolbox pane when a GUI action, component, or function library is in focus in the document pane.

Add a step using a test object

Drag the test object into the action or component. A step with the default operation for that test object is inserted. Modify the operation and enter arguments for the step, as needed.

You can drag test objects into an action or scripted component in both the Keyword View and the Editor.

Add a call to a function

Drag the function into the action, component, or function library. A comment and call to that function are added.

The comment indicates that a call to the function was added to your action or component, and indicates any necessary arguments. You then provide the arguments for that function.

You can drag functions into an action or scripted component in both the Keyword View and the Editor.

Open the test object's object repository

Do one of the following:

- Right-click the test object and select **Open Resource**.
- Double-click the test object.

The Object Repository window opens and highlights that test object.

Locate a function in an action or function library

Double-click the function you want to locate, or right-click it and select **Open Resource**. The action or associated function library containing that function opens in the document pane and points to the function.

Copy and paste the selected test object or function

Right-click the test object or function and select **Copy Keyword**. The test object or function is now in the clipboard, and you can paste it into a function library, or for tests, into an action in the Editor.

Pasting the test object or function into the document pane has the same result as dragging it in from the Toolbox pane.

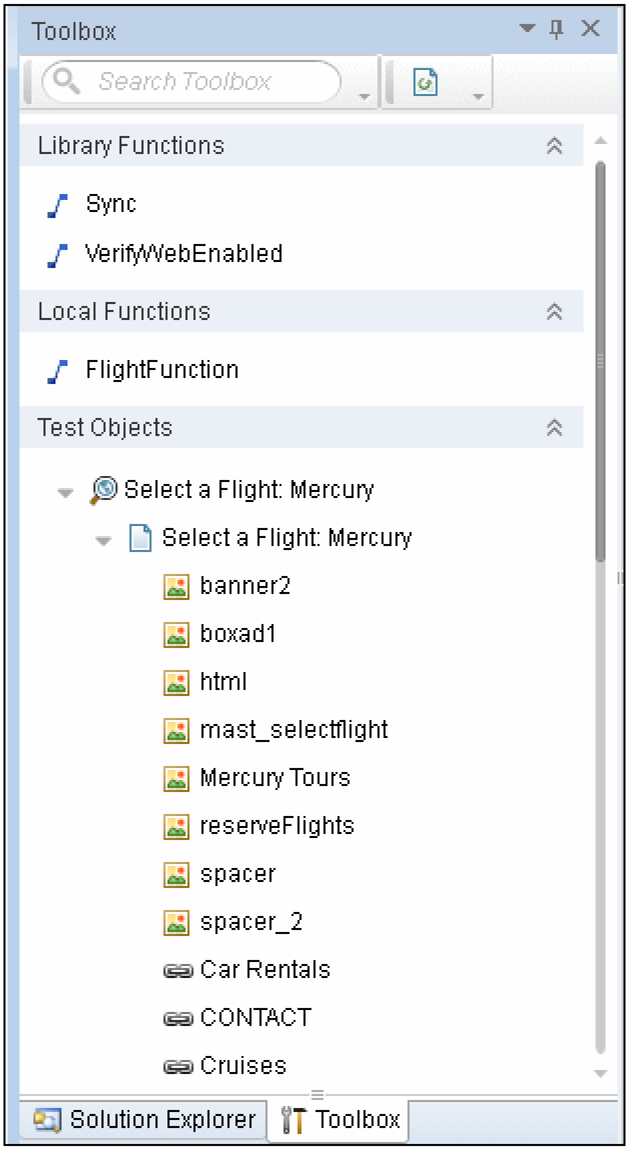
References

Toolbox Pane User Interface (GUI Testing)

For details about the API testing Toolbox pane, see "Toolbox Pane User Interface (API Testing) " on page 514. For details about the business process testing Toolbox pane, see "Toolbox Pane User Interface (BPT in UFT)" on page 518.

Relevant for: GUI actions, components, and function libraries

The Toolbox pane displays test objects and functions available to your document sorted alphabetically. You can view these items, open them in the object repository, action, or function library in which they are defined, or use them to add steps to your document (by dragging or copying the item into the document pane).



To access	1. Do one of the following: ▪ Ensure that an action, component, or function library 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 **View > Toolbox**. **Note:** If the Toolbox pane is open, but not in focus, click the **Toolbox** tab at the bottom.

Important information	• Checkpoint and output value objects are not displayed in this pane, even if they are included in an associated object repository. • The Toolbox pane displays the test objects and functions available to the document that is currently in focus. • For function libraries, the Toolbox pane displays only the **Local Functions** node.
Relevant tasks	• "How to Work with the Toolbox Pane (GUI Testing) " on page 508 • "How to Select an Item for Your Step " on page 926 • "How to Create a GUI Test Using the Keyword-Driven Methodology" on page 836
See also	For details on dragging and dropping test objects from other locations, see: • "Object Repository Window" on page 1274 • "How to Manage Objects in Shared Object Repositories" on page 1294

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

UI Elements	Description
▼ ▶	**Expand/Collapse sub tree.** Expands or collapses the **Library Functions**, **Local Functions**, or **Test Objects** nodes.
Library Functions	All functions available to the action or component in associated function libraries. • The functions are sorted alphabetically, regardless of the function library in which they are defined. • When you click on a function, the file name of its function library is displayed in the description area at the bottom of the pane. • When you hover over a function, the file name of its function library is displayed in a tooltip. **Note:** • If more than one function in the same function library has the same name, only the first function is displayed (this is the one UFT uses). • If functions in different function libraries have the same name, they are displayed in order of the priority of their function libraries.

UI Elements	Description
Local Functions	All of the functions defined within the action or function library, sorted alphabetically. **Note:** If more than one function has the same name, only the first is displayed (this is the one UFT uses). **Available for:** actions, scripted components, function libraries.
Test Objects	All test objects available to the action or component. This includes test objects located in the local object repository of the action or component, or in shared object repositories associated with the action or the component's application area. • The test objects are grouped by their parent objects and their test object type, and sorted alphabetically, regardless of the object repository in which they are stored. • When you click on a test object, its object repository's file path is displayed in the description area at the bottom of the pane. • When you hover over a test object, its object repository's file path is displayed in a tooltip. **Note:** If the same test object exists in more than one object repository, only one test object is displayed (the one from the object repository with the highest priority).
<Search Toolbox>	A text box in which you enter the text by which you want to filter test objects and functions. **Note:** For test objects, the filter applies only to the child node and not to any of its parent test objects.
↻	Refreshes the Toolbox pane.
Open Resource	(**Context-menu**) For test objects, this opens the Object Repository window to display the resource file in which the test object is stored. You can also activate this option by double-clicking a test object. For functions, this opens the containing action or function library in the document pane, and points to the function.
Copy Keyword	(**Context-menu**) Copies the selected keyword to the Clipboard. The test object or function is now in the clipboard, and you can paste it into an action, function library, or scripted component in the Editor.

UI Elements	Description
<description>	The bottom area of the Toolbox pane displays the name of the selected test object or function, and the location or resource in which it is stored.

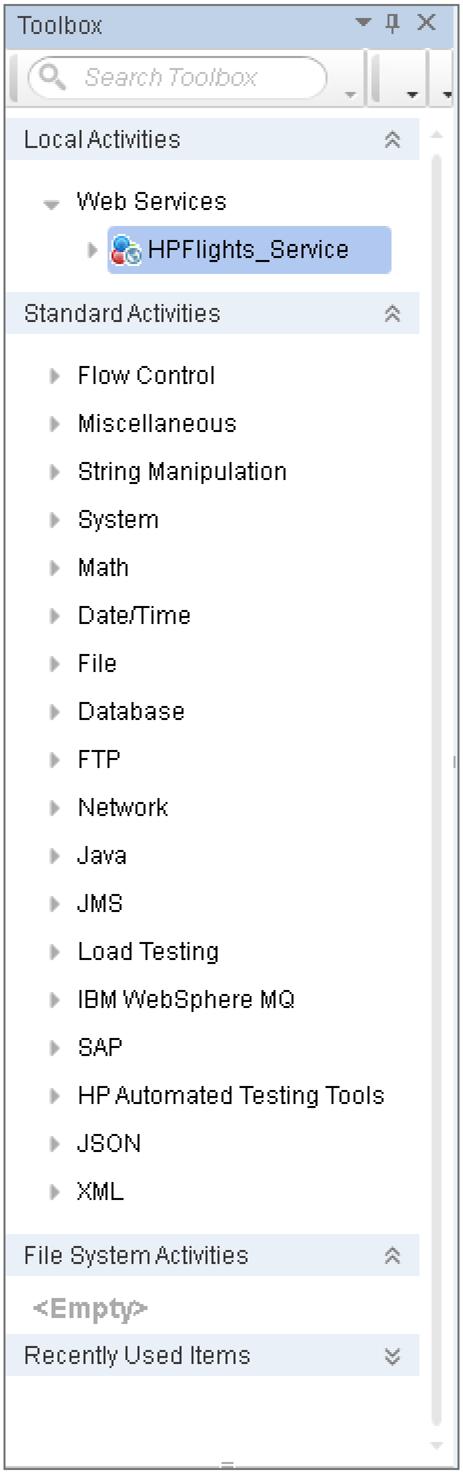
Toolbox Pane User Interface (API Testing)

For details about the GUI testing Toolbox pane, see "Toolbox Pane User Interface (GUI Testing)" on page 510. For details about the business process testing Toolbox pane, see "Toolbox Pane User Interface (BPT in UFT)" on page 518.

Relevant for: API actions, tests, and components

The Toolbox pane enables you to view all of the available activities. This includes imported services and built-in activities such as that you can use in your visual design, such as **Replace String** or **File Exists**.

This pane is UFT's main window for creating and populating tests.



To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. Do one of the following: ▪ Click the **Toolbox** tab in the bottom of the left pane. ▪ Select **View > Toolbox** or click the Toolbox tab in the main UFT window.
Important information	• You double-click or drag activities from this pane onto the canvas. • The activity becomes as test step when it appears in the canvas. • Click on the down arrows at the top of the pane to access the toolbar controls.
Relevant tasks	"How to Create an API Test" on page 1598

Toolbar Controls

The following toolbar controls elements are included in the Toolbox pane. To see all of the controls, expand the pane or use the drop down menus.

Tree Elements	Description
	Expand All. Expands all nodes in the Toolbox pane.
	Collapse All. Collapses all nodes in the Toolbox pane.
	Security Settings. Opens the "Security Settings for Port <Port_Name> Dialog Box" (described on page 1900). These settings apply to all operations in the port. **Note:** Only available when selecting the Port node of a Web service.
	Choose Activities. Opens the Choose Toolbox Activities dialog box, for selecting activities stored in the File System or ALM repository to add to the Toolbox pane. This is useful for retrieving activities that you removed from the Toolbox using the **Remove Activities** command.

Tree Elements	Description
	Remove Activities. Opens the Remove Toolbox Activities dialog box for selecting the activities to remove from the Toolbox pane. **Note:** • Removing an activity stored in the repository, only removes it from the Toolbox pane. It remains, however, in the repository, for future use. • This command does not apply to **Standard Activities**.
↻	**Update WSDL.** Reimports the WSDL from its original location. If the service's operations changed, this will be reflected in its node in the Toolbox pane. **Note:** Only available when selecting the parent node of a Web service.
Update WSDL from	Opens the Update WSDL From dialog box, allowing you to update the WSDL from any location. **Note:** Only available when selecting the parent node of a Web service.
	Refresh. Refreshes the activities in their stored location. This is useful when the WSDL is stored on a shared location and may have been modified by another user. You can refresh the activities instead of re-importing the service. **Note:** Only available when selecting the parent node of a Web service.
✖	**Delete.** Deletes the selected entry from the Toolbox pane. This is not available for built-in activities.
✓	**Validate WSI-Compliance.** Runs the WSI validator. The Output pane shows a summary of the validation and provides a path to the report. **Note:** Only available when selecting the parent node of a Web service.
	Edit Service. Opens the "Add/Edit REST Service Dialog Box" (described on page 1772). **Note:** Only available when selecting the parent node of a REST service.

Tree Elements	Description
Move to	**Move to.** An expandable menu that enables you to move an item to the file system or ALM repository. For details, see "Activity Sharing" on page 1737. **Note:** Only available when selecting the parent node of an item in the **Local Activities** section.
Filter box	Filters the Toolbox pane display by the entered text.

Toolbox Activities

The Toolbox pane contains the following sections:

Activity Category	Description
Local Activities	A tree hierarchy of all imported or items created by the user: **Web Services**, **REST Services**, and **.NET Assemblies**. For details, see "Local Activities" on page 1733.
Standard Activities	The built-in activities, by category. For details, see "Standard Activities" on page 1611.
File System Activities	Local activities that were moved to the file system repository using the **Move to** option from the right-click menu. For details, see "Activity Sharing" on page 1737.
ALM Activities	Local activities that were moved to the ALM repository using the **Move to** option from the right-click menu. For details, see "Activity Sharing" on page 1737.
Recently Used Items	The activities or operations that were used most recently.

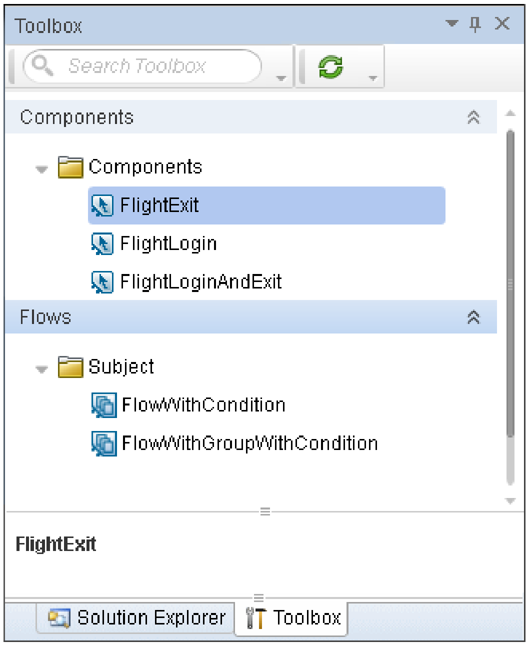
Toolbox Pane User Interface (BPT in UFT)

For details about the GUI testing Toolbox pane, see "Toolbox Pane User Interface (GUI Testing)" on page 510. For details about the API testing Toolbox pane, see "Toolbox Pane User Interface (API Testing) " on page 514.

Relevant for: business process tests and flows

The Toolbox pane enables you to view all of the available components and flows in the project that you can use in your business process test or flow.

This pane is UFT's main window for creating and populating business process tests and flows.



To access	1. Make sure that an business process test or flow is in focus in the document pane. 2. Do one of the following: ▪ Click the **Toolbox** tab in the bottom of the left pane. ▪ Select **View > Toolbox** or click the Toolbox button in the main UFT window.
Important information	• Components and flows are organized according to their ALM paths. • Double-click or drag and drop components or flows from this pane into the test or flow displayed in the document pane. • When you select an item in the tree, its name is displayed in the description area at the bottom of the pane.
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
⌄ ⌃	**Expand/Collapse sub tree.** Expands or collapses the **Components** or **Flows** nodes.

UI Element	Description
<Search Toolbox>	A text box that enables you to filter the components and flows displayed by the entered text.
🔄	Refreshes the content of the toolbox pane with any new components or flows that were recently created in ALM.

Part 3: UFT Configuration

Chapter 19: UFT Global Options

Relevant for: GUI testing and API testing

This chapter includes:

Concepts

Global Options - Overview

Relevant for: GUI testing and API testing

The Options dialog box enables you to modify the general appearance and behavior of UFT. For example, you can define the user interface language, set startup options, or modify the font and colors of code elements in the Editor. The values you set remain in effect for all documents and for subsequent testing sessions.

> **Note:** The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.

For user interface details, see:

- "General Tab (Options Dialog Box)" on the next page
- "GUI Testing Tab (Options Dialog Box)" on page 535
- "API Testing Tab (Options Dialog Box)" on page 564
- "Coding Tab (Options Dialog Box)" on page 572
- "Text Editor Tab (Options Dialog Box)" on page 574

Reference

General Tab (Options Dialog Box)

Relevant for: GUI tests and components and API testing

This tab enables you to define general options for UFT.

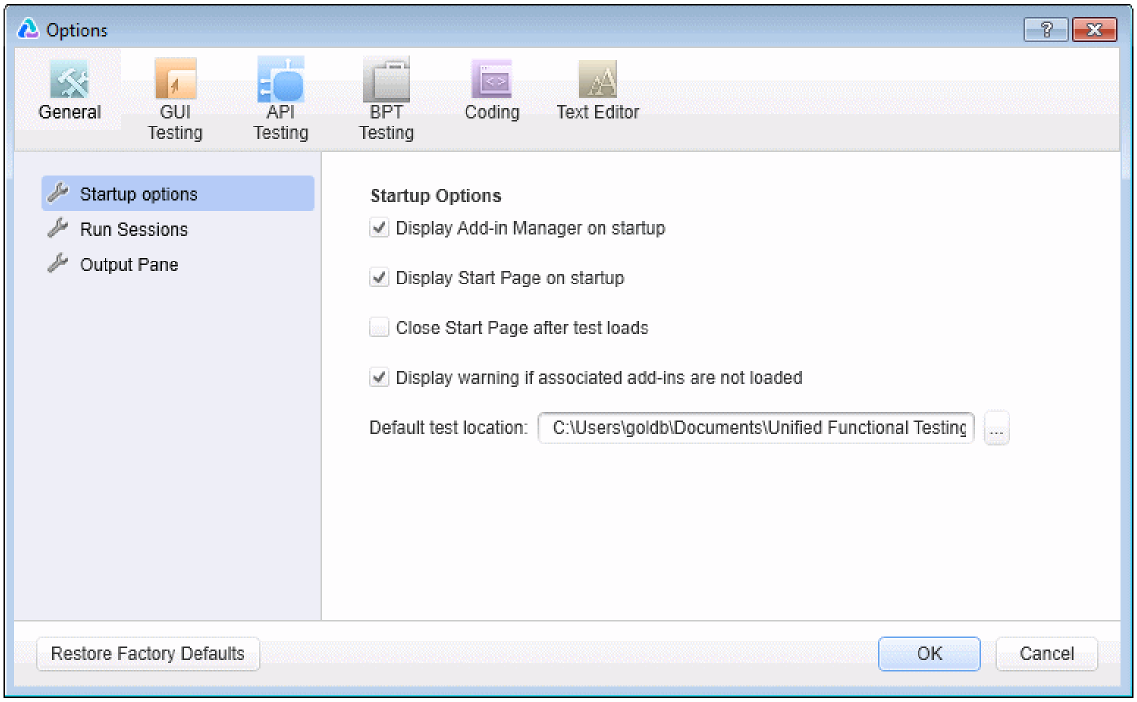
To access	Select **Tools > Options > General** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "GUI Testing Tab (Options Dialog Box)" described on page 535 • "API Testing Tab (Options Dialog Box)" (described on page 564 • "Coding Tab (Options Dialog Box)" (described on page 572) • "Text Editor Tab (Options Dialog Box)" (described on page 574)

This tab includes the following panes:

Startup Options (Options Dialog Box > General Tab)

Relevant for: GUI tests and components and API testing

This pane enables you to select what to show when UFT opens.



Startup Options

☑ Display Add-in Manager on startup

☑ Display Start Page on startup

□ Close Start Page after test loads

☑ Display warning if associated add-ins are not loaded

Default test location: C:\Users\goldb\Documents\Unified Functional Testing

To access	Select **Tools > Options > General** tab > **Startup Options** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
Relevant tasks	"How to Start UFT" on page 114
See also	• "Start Page" on page 121 • "Add-in Manager Dialog Box" on page 118

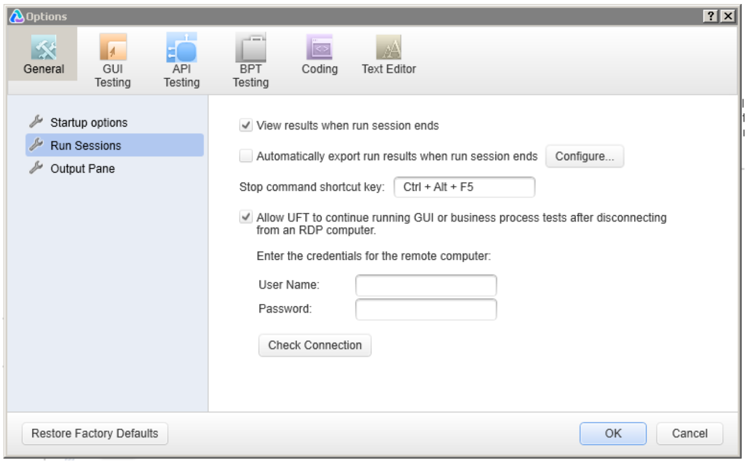
User interface elements are described below:

UI Element	Description
Display Add-in Manager on startup (GUI testing only)	Opens the Add-in Manager when starting UFT. For details, see the section on loading add-ins in the *HP Unified Functional Testing Add-ins Guide*. **Note:** The add-ins selected in the Add-in Manager are relevant only for GUI tests and components.
Display Start Page on startup	Displays the Start Page when UFT opens.
Close Start Page after test loads	Instructs UFT to close the Start Page after a test or business component is created or opened.
Display warning if associated add-ins are not loaded (GUI testing only)	Displays a warning message if you open a test or business component that requires add-ins that are not currently loaded.
Default test location	The default place to save a new test. **Default:** C:\My Documents\Unified Functional Testing **Note:** Business components are always saved in the Business Components module in ALM.

Run Sessions Pane (Options Dialog Box > General Tab)

Relevant for: GUI tests and components and API testing

This pane enables you to determine global UFT run settings that are applicable to both GUI testing and API testing.



☑ View results when run session ends

□ Automatically export run results when run session ends

Stop command shortcut key: Ctrl + Alt + F5

☑ Allow UFT to continue running GUI or business process tests after disconnecting from an RDP computer.

Enter the credentials for the remote computer:

User Name:

Password:

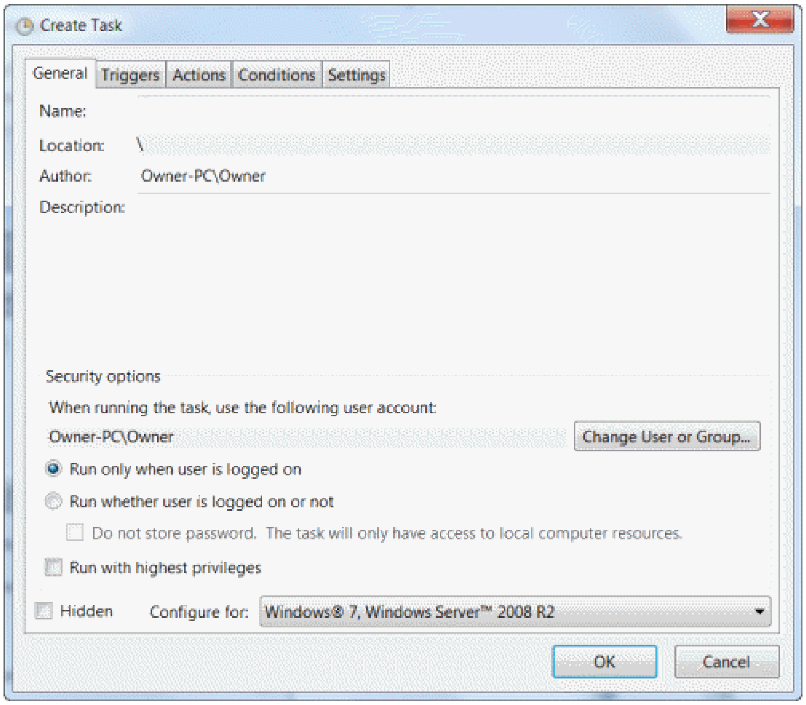
To access	Select **Tools > Options > General** tab > **Run Sessions** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
Relevant tasks	See the following tasks in the *HP Run Results Viewer User Guide*: • How to Open Run Results • How to Manually Submit Defects to ALM • How to Automatically Submit Defects to an ALM Project • How to Export Run Results)
See also	The following topics in the *HP Run Results Viewer User Guide*: • The Run Results Viewer overview • The section on the Run Results Viewer User Interface

User interface elements are described below:

UI Element	Description
View results when run session ends	Instructs UFT to open the Run Results Viewer and display the results automatically following the run session.

UI Element	Description
Automatically export run results when run session ends	Instructs UFT to export the run results to the location you specify in the "Configure Automatic Export of Run Results Dialog Box" (described on page 531). **Note:** If this option is selected, and the Run Results Viewer is not installed on the computer running the test or component, then the results are not exported and an error message is displayed at the end of the run session.
Configure	Opens the "Configure Automatic Export of Run Results Dialog Box" (described on page 531), in which you can define settings that instruct UFT to automatically export run results to a specific folder in the file system after a run session. These settings specify the type of report, the export location in the file system, and whether to export all run results or only results for failed run sessions.
Stop command shortcut key	Enables you to define a shortcut key or key combination that stops the current record (for GUI tests only) or run operation, even if UFT is not in focus or is in hidden mode. Click in the field and then press the required key or key combination on the keyboard. The default key combination is **CTRL+ALT+F5**. **Note:** It is important to define a shortcut that is not already defined for some other operation by application being tested. If this is the case and: • you open the application manually before you click **Record** or **Run**, the shortcut defined in the application is applied for its original purpose. • you start a record or run session and UFT opens the application for you, the shortcut you define stops the session.

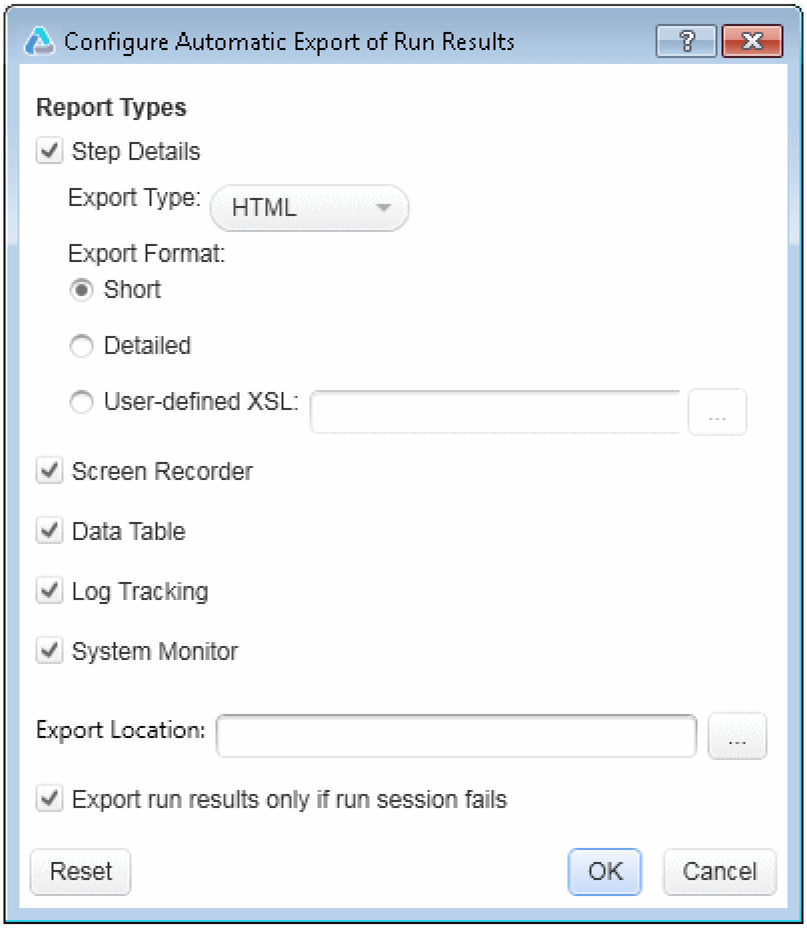
UI Element	Description
Allow UFT to continue running GUI or business process tests after disconnecting from an RDP computer	Enables GUI or business process tests in UFT to continue running on a remote computer when a local RDP (Remote Desktop Protocol)-based client is disconnected from it. **Note:** If you want to run UFT in a minimized RDP session, and you are using an RDP 6.0 or later client, you can enable it by setting a registry value on the computer that is running the RDP client: 1. If it does not exist, create the RemoteDesktop_SuppressWhenMinimized registry value (DWORD type) in one of the following registry paths on the computer that is running the RDP client: ▪ **For 32-bit operating systems:** <HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE>\Software\Microsoft\Terminal Server Client ▪ **For 64-bit operating systems:** <HKEY_CURRENT_USER>\Software\Wow6432Node\Microsoft\Terminal Server Client 2. Set the data for this value to 2. You must restart your remote session in order for this setting to take effect. In order to run a test with a disconnected Remote Desktop connection, ensure the following: • You remain logged on to the computer running UFT. • If you want to use automation to run a GUI test and schedule it in the Windows Task Scheduler, the **Run only when user is logged on option** should be selected.

UI Element	Description
	 Create Task General | Triggers | Actions | Conditions | Settings Name: Location: \ Author: Owner-PC\Owner Description: Security options When running the task, use the following user account: Owner-PC\Owner [Change User or Group...] ● Run only when user is logged on ○ Run whether user is logged on or not □ Do not store password. The task will only have access to local computer resources. □ Run with highest privileges □ Hidden Configure for: Windows® 7, Windows Server™ 2008 R2 OK Cancel • For a UFT server that allows multiple users to log in with the same username/password: When a user initiates a test run and disconnects from RDP during the run, the run execution goes to the background and the user has no access to the execution when they log in again, because the test is executed in another session. **Note:** This feature is not supported in the Microsoft Windows® XP environment or the Hyper-V virtualization server.
User Name	The user name that the local RDP-based client uses to connect to the remote computer running UFT, in the following format: <domain>\<user Name>.
Password	The password that the local RDP-based client uses to connect to the remote computer running UFT.
Check Connection	Verifies the connection to the remote computer running UFT, using the user name and password that you entered in the fields above.

Configure Automatic Export of Run Results Dialog Box

Relevant for: GUI testing and API testing

This dialog box enables you to define settings that instruct UFT to automatically export run results to a specific folder in the file system after a run session.



Configure Automatic Export of Run Results

Report Types

☑ Step Details

Export Type: HTML

Export Format:

● Short

○ Detailed

○ User-defined XSL:

☑ Screen Recorder

☑ Data Table

☑ Log Tracking

☑ System Monitor

Export Location:

☑ Export run results only if run session fails

To access	1. Select **Tools > Options > General** tab > **Run Sessions** node to open the Run Sessions pane. 2. Select **Automatically export run results when run session ends.** 3. Click the **Configure** button.

Important information	• **GUI testing only: Screen capture images are not exported for steps on Web-based applications.** When you export run results containing steps on a Web application, any screen capture images for those steps are not exported to the file. This is because for Web-based applications, the Run Results Viewer displays the HTML corresponding to the relevant Web page (with downloaded images) rather than a captured image and thus no image is saved with the report. • **Manually exporting run results.** You can also export run results manually after a particular run session. For details, see the section on how to export run results (described in the *HP Run Results Viewer User Guide*). • **Note for API testing:** Not all options are relevant for API tests. • The settings from this dialog box are saved in a configuration file, ReportExportOption.xml.
See also	"Test Runs Pane (Options Dialog Box > GUI Testing Tab)" on page 539

User interface elements are described below:

UI Element	Description
Step Details	The main report type. This includes the entire run results tree. (Default) • **Export type.** The file type for the exported results. Possible options: ▪ **HTML** (Default) ▪ **PDF** • **Export format.** The format and amount of detail to include in the exported results. Possible options: ▪ **Short.** Exports a summary line (when available) for each item in the run results tree. The short report does not include still images associated with the steps in your run results. (Default) ▪ **Detailed.** Exports all available information for each item in the run results tree. The detailed report includes still images associated with the steps in your run results. (In the Run Results Viewer, these images are displayed in the Captured Data pane.) If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included in the printed report. ▪ **User-defined XSL.** Enables you to browse to and select a customized .xsl file. You can create a customized .xsl file that specifies the information to be included in the exported report, and the way it should appear. For details, see the section on the Run Results XML File (described in the *HP Run Results Viewer User Guide*).

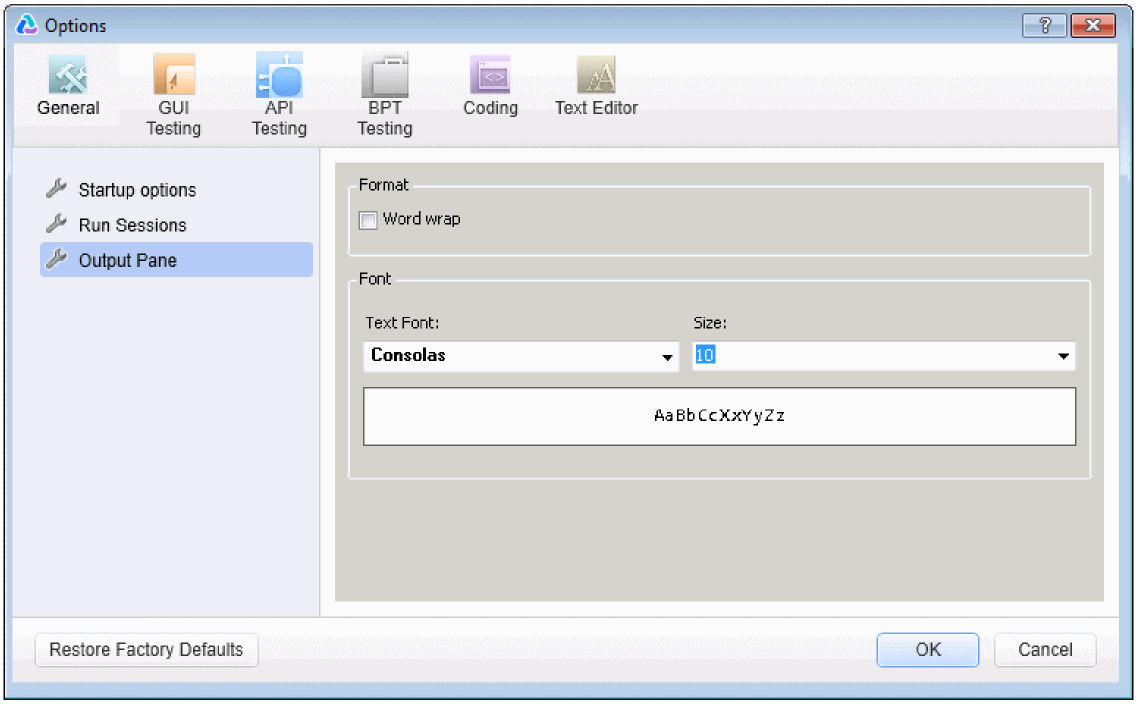
UI Element	Description
Screen Recorder (GUI testing only)	A movie of the run session. For details on how to configure when and how UFT captures screens of the application being tested, see "Screen Capture Pane (Options Dialog Box > GUI Testing Tab)" on page 557.
Data Table	The runtime version of the data table associated with your ALM configuration. It displays the values used to run a test or configuration that contains Data Table parameters, as well as any output values retrieved from a test or an ALM configuration during a run session. For details on the GUI testing Data pane, see "Data Pane" on page 221. For details on theAPI testing Data pane, see "Data Pane User Interface (API Testing)" on page 252.
Log Tracking (GUI testing only)	A complete list of log messages that UFT received from your application during the run session. For details on what is displayed, see the section on the Run Results Viewer Log Tracking Pane (described in the *HP Run Results Viewer User Guide*). For configuration details, see "Log Tracking" on page 586.
System Monitor (GUI testing only)	The system counters that are monitored for a run session displayed in a line graph. For configuration details, see "Local System Monitor" on page 585.
Export location	The folder in which to store the exported files. Clicking the browse button opens the Browse For Folder dialog box, enabling you to select a folder on the file system. If you do not specify a folder, UFT automatically exports the files to the default folder: <test folder>\<result folder>\Report. (For example, %ProgramFiles%\HP\UFT\Tests\MyTest\Res9\Report.)
Export run results only if the run session fails	• Select this check box to export run results only for failed run sessions. • Clear this check box to export all run session results.

UFT stores the export options in the ReportExportOption.xml file in the %AppData%\Roaming\Hewlett-Packard\UFT\ConfigurationUI folder. This file is useful when using the Test Batch Runner from the command line. For details, see "How to Run a Test Batch Using the Windows Command Line" on page 648.

Output Pane (Options Dialog Box > General Tab)

Relevant for: GUI tests and components and API testing

This pane enables you to define display options for the Output pane.



□ Word wrap

Font

Text Font: Consolas

Size: 10

AaBbCcXxYyZz

To access	Select **Tools > Options > General** tab > **Output Pane** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "Output Pane Overview" on page 380 • "Output Pane User Interface" on page 381

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

UI Element	Description
Word wrap	When selected, wraps the text of each message onto the next line.
Text Font	The font in which to display the content of the Output pane.
Size	The size in which to display the text in the Output pane.
<preview area>	Shows an example of the formatting selected in the **Text Font** and **Size** drop-down lists.

GUI Testing Tab (Options Dialog Box)

Relevant for: GUI tests and components

This tab enables you to modify global testing options for GUI tests and components.

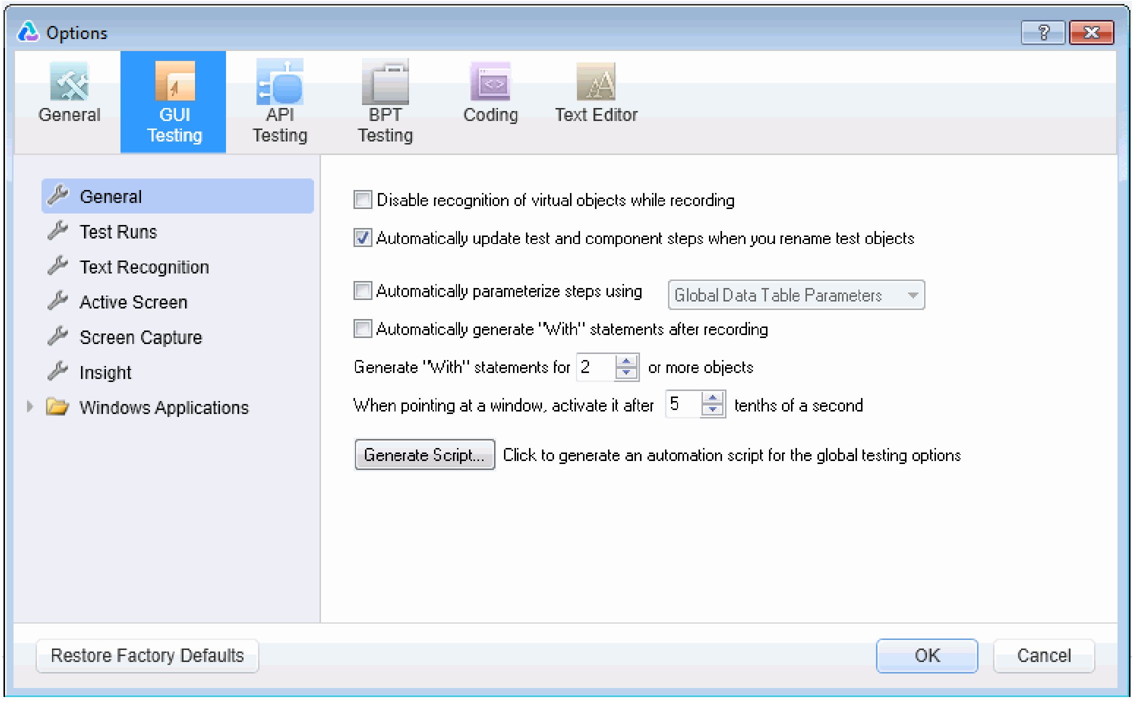
To access	Select **Tools > Options > GUI Testing** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. This pane may also contain additional nodes, depending on the add-ins that are currently loaded with a test or component. For details on add-ins, see the relevant section in the *HP Unified Functional Testing Add-ins Guide*.
See also	• "General Tab (Options Dialog Box)" on page 524 • "API Testing Tab (Options Dialog Box)" on page 564 • "Coding Tab (Options Dialog Box)" on page 572 • "Text Editor Tab (Options Dialog Box)" on page 574

This tab includes the following panes:

General Pane (Options Dialog Box > GUI Testing Tab)

Relevant for: GUI tests and components

This pane contains general options for working with GUI tests and components.



□ Disable recognition of virtual objects while recording

☑ Automatically update test and component steps when you rename test objects

□ Automatically parameterize steps using Global Data Table Parameters

□ Automatically generate "With" statements after recording

Generate "With" statements for 2 or more objects

When pointing at a window, activate it after 5 tenths of a second

Click to generate an automation script for the global testing options

To access	Select **Tools > Options > GUI Testing** tab > **General** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.

User interface elements are described below:

UI Element	Description
Disable recognition of virtual objects while recording (GUI tests only)	Determines whether the defined virtual objects stored in the Virtual Object Manager are recognized while recording. For details, see "Virtual Objects" on page 1384.

UI Element	Description
Automatically update test and component steps when you rename test objects	Determines whether to automatically update test and component steps when you rename test objects in the local or shared object repository. For details, see "Renaming Test Objects " on page 1206. **Note:** Applicable only if there are no syntax errors in the test or component when you rename the test object.
Automatically parameterize steps using **(tests only)**	Instructs UFT to automatically parameterize the test object method arguments in any action in which you recorded one or more steps during a recording session. You can select to parameterize the steps with **Global Data Table Parameters** or **Test Parameters**. For details, see "Automatically Parameterizing Steps" on page 1539.
Automatically generate "With" statements after recording **(tests only)**	Instructs UFT to automatically generate With statements when you record. For details, see "How to Generate With Statements for Your Test" on page 978.
Generate "With" statements for __ or more objects **(tests only)**	Indicates the minimum number of identical, consecutive objects for which you want to apply the With statement. This setting is used when UFT automatically generates With statements after recording and when you select to generate With statements for an existing action. **Default = 2** For details, see "How to Generate With Statements for Your Test" on page 978.
When pointing at a window, activate it after __ tenths of a second	Specifies the time (in tenths of a second) that UFT waits before it sets the focus on an application window when using the pointing hand to point to an object in the application (for example, when using the Object Spy, checkpoints, Step Generator, Recovery Scenario Wizard, and so forth). **Default = 5**

UI Element	Description
Generate Script	Generates a UFT script containing most of the current global testing options. When you click the **Generate Script** button, a Save As dialog box opens, enabling you to specify the name and file system location to store the generated file. You can use some or all of the script lines from this generated script in a test. This can be useful, for example, if you want to create an initialization script that opens UFT with a pre-defined set of options applied. For details, see "UFT Automation Scripts" on page 1155 and the *UFT Object Model Reference* (**Help > HP Unified Functional Testing Help > Object Model Reference for GUITesting**).

Sample Generated Script for Global Testing Options

Relevant for: GUI tests and components

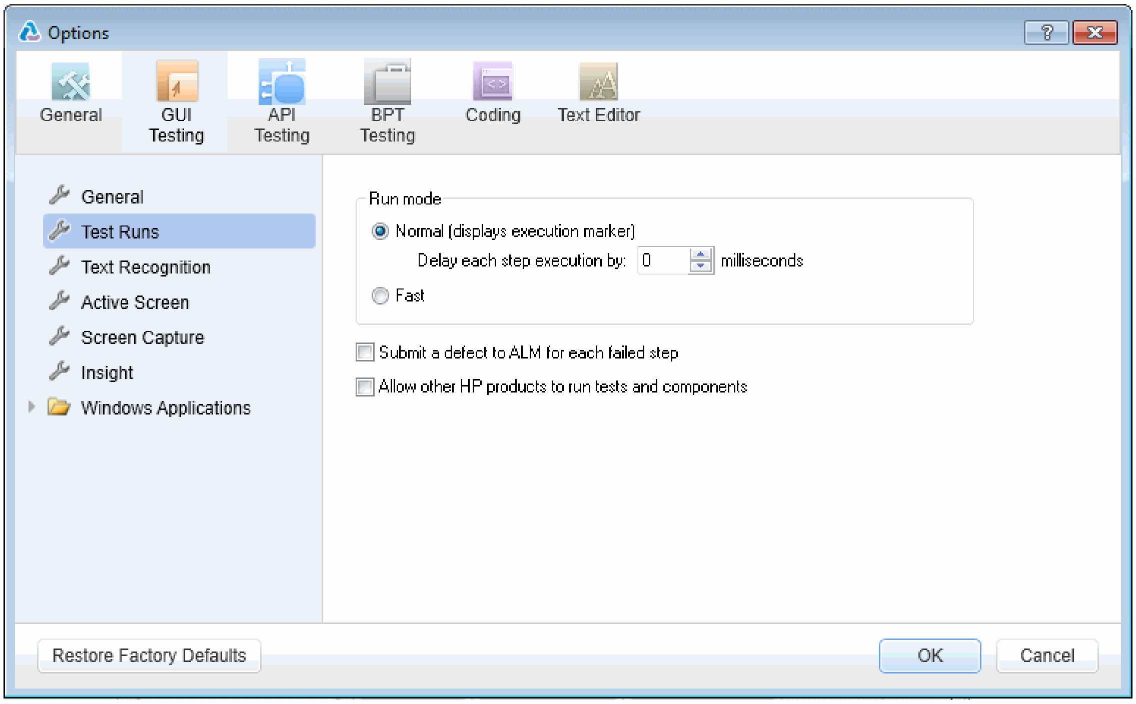
The UFT script below shows a sample script that was generated using the Generate Script button in the Options dialog box. The script was generated with the Web and Java Add-ins loaded.

```
Dim App 'As Application
Set App = CreateObject("QuickTest.Application")
App.Launch
App.Visible = True
App.Options.DisableVORecognition = False
App.Options.AutoGenerateWith = False
App.Options.WithGenerationLevel = 2
App.Options.TimeToActivateWinAfterPoint = 500
App.Options.SaveLoadAndMonitorData = True
App.Options.Run.RunMode = "Normal"
App.Options.Run.ViewResults = True
App.Options.Run.StepExecutionDelay = 0
App.Options.Run.MovieCaptureForTestResults = "Never"
App.Options.Web.AddToPageLoadTime = 10
App.Options.Web.RecordCoordinates = False
App.Options.Web.RecordMouseDownAndUpAsClick = False
App.Options.Web.RecordAllNavigations = False
App.Options.Web.RecordByWinMouseEvents = ""
App.Options.Web.BrowserCleanup = False
App.Options.Web.RunOnlyClick = False
App.Options.Web.RunMouseByEvents = True
App.Options.Web.RunUsingSourceIndex = True
App.Options.Web.EnableBrowserResize = True
App.Options.Web.PageCreationMode = "URL"
App.Options.Web.CreatePageUsingUserData = "Get Post"
App.Options.Web.CreatePageUsingNonUserData = ""
App.Options.Web.CreatePageUsingAdditionalInfo = True
App.Options.Web.FrameCreationMode = "URL"
App.Options.Web.CreateFrameUsingUserData = "Get Post"
App.Options.Web.CreateFrameUsingNonUserData = ""
App.Options.Web.CreateFrameUsingAdditionalInfo = True
App.Options.Java.RecordListByIndex = False
App.Options.Java.RecordComboByIndex = False
App.Options.Java.RecordTreeByIndex = False
App.Options.Java.RecordTabByIndex = False
App.Options.Java.AWTEventModel = "Auto"
App.Options.Java.AnalogTableRecording = False
App.Options.Java.TreePathSeparator = ";"
App.Options.Java.TableExternalEditors = ""
App.Options.Java.TableInternalEditors = ""
App.Options.WindowsApps.AttachedTextRadius = 35
App.Options.WindowsApps.AttachedTextArea = "TopLeft"
App.Options.WindowsApps.ExpandMenuToRetrieveProperties = True
App.Options.WindowsApps.NonUniqueListItemRecordMode = "ByName"
App.Options.WindowsApps.RecordOwnerDrawnButtonAs = "PushButtons"
App.Options.WindowsApps.ForceEnumChildWindows = 0
App.Options.WindowsApps.ClickEditBeforeSetText = 0
App.Options.WindowsApps.VerifyMenuInitEvent = 0
App.Options.TextRecognitionOrder = "APIThenOCR"
App.Options.TextRecognitionBlockType = "Multiple"
App.Options.TextRecognitionLanguages = "English"
App.Options.DisplayKeywordView = True
App.Folders.RemoveAll
App.Folders.Add("C:\Temp\MyTests")
```

Test Runs Pane (Options Dialog Box > GUI Testing Tab)

Relevant for: GUI tests only

This pane enables you to determine how UFT runs GUI tests and components. For additional run options, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 526.



Run mode

● Normal (displays execution marker)

Delay each step execution by: 0 milliseconds

○ Fast

□ Submit a defect to ALM for each failed step

□ Allow other HP products to run tests and components

To access	Select **Tools > Options > GUI Testing** tab > **Test Runs** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
Relevant tasks	"How to Run an API Test" on page 643
See also	• "Running API Tests - Overview " on page 636 • "Run Dialog Box" on page 651

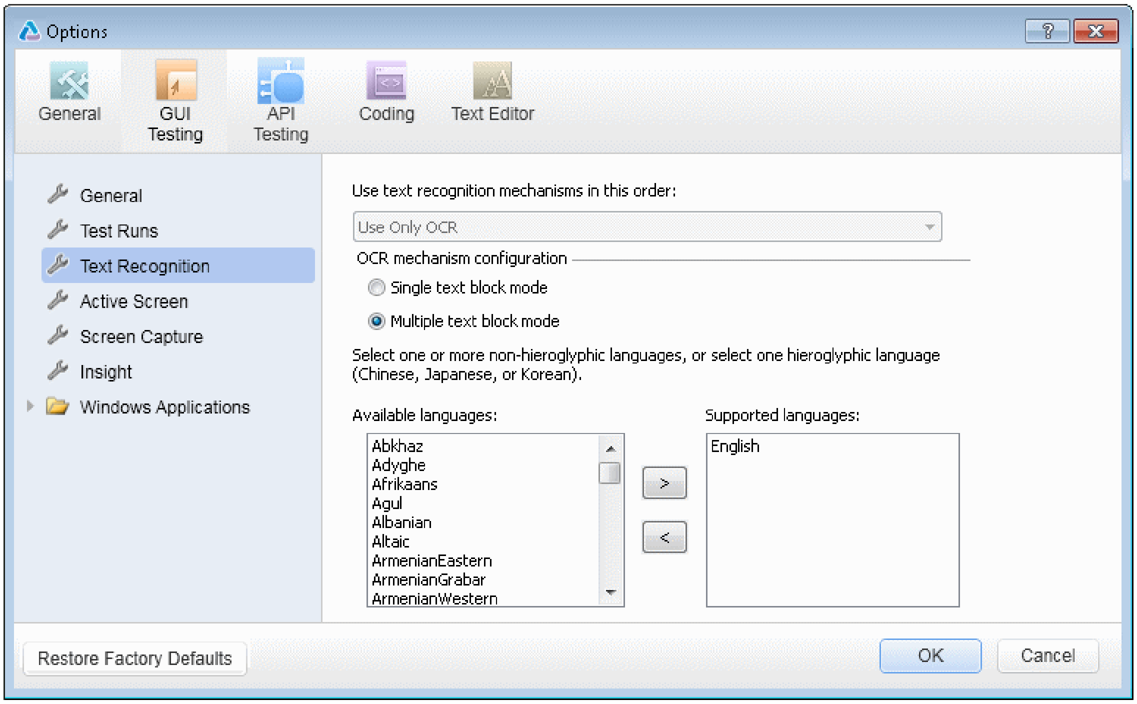
User interface elements are described below:

UI Element	Description
Run mode	Instructs UFT how to run your test or component: • **Normal (displays execution marker) (default).** Runs your test or component with the execution arrow to the left of each step or statement as it is performed. If the test contains multiple actions, the tree in the Keyword View **Item** column expands to display the steps, and the Editor displays the script, of the currently running action. **Delay each step execution by.** You can specify the time in milliseconds that UFT should wait before running each consecutive step (up to a maximum of 10000 ms.). The **Normal** run mode option requires more system resources than the **Fast** option, described below. **Note:** You must have Microsoft Script Debugger installed to enable this mode. For details, see the *HP Unified Functional Testing Installation Guide*. • **Fast.** Runs your test or component without the execution arrow to the left of the Keyword View or Editor (for tests) and does not expand the item tree or display the script of each action as it runs. This option requires fewer system resources. **Note:** When running a test set from ALM, tests or components are automatically run in **Fast** mode, even if **Normal** mode is selected.
Submit a defect to ALM for each failed step	Instructs UFT to automatically submit a defect to ALM for each failed step in your test. This option is available only when you are connected to an ALM project. For details, see the section on how to automatically submit defects to an ALM project in the *HP Run Results Viewer User Guide*.
Allow other HP products to run tests and components	Enables other HP products such as ALM and Test Batch Runner to run tests and components on this computer. **Note:** If the **Windows Firewall** is **turned on** on your computer, and you want to enable business process tests to be run on your computer from a remote ALM client, then you must also manually create a firewall exception for the bp_exec_agent.exe remote agent. For details, see "How to Enable the BPT Remote Agent" on page 733.

Text Recognition Pane (Options Dialog Box > GUI Testing Tab)

Relevant for: GUI tests and components

This pane enables you to configure how UFT identifies text in your application. You can use this pane to modify the default text capture mechanism, OCR (optical character recognition) mechanism mode, and the language dictionaries the OCR mechanism uses to identify text.



Use text recognition mechanisms in this order: Use Only OCR

OCR mechanism configuration

○ Single text block mode

● Multiple text block mode

Select one or more non-hieroglyphic languages, or select one hieroglyphic language (Chinese, Japanese, or Korean).

Available languages:

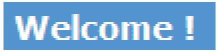
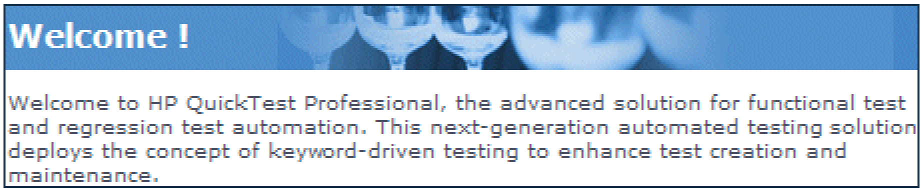
Abkhaz
Adyghe
Afrikaans
Agul
Albanian
Altaic
ArmenianEastern
ArmenianGrabar
ArmenianWestern

Supported languages:

English

To access	Select **Tools > Options > GUI Testing** tab > **Text Recognition** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
Related tasks	• "How to Configure Text Recognition Settings" on page 1429 • "How to Insert a Checkpoint in a GUI Test or Component" on page 1419 • "How to Create or Modify an Output Value Step" on page 1494
See also	• "Text Recognition Overview" on page 1408 • "Guidelines for Text Recognition" on page 1409 • "Checking Text in an Image - Use-Case Scenario" on page 1414 • "Checking Text Overview " on page 1408

User interface elements are described below:

UI Element	Description
Use text recognition mechanisms in this order	Specifies the text recognition mechanism that UFT uses when capturing text. **Possible values:** - **First Windows API then OCR.** (Default) Instructs UFT to first try to retrieve text directly from the object using the Windows API-based mechanism. If no text can be retrieved (for example, because the text is part of a picture), UFT tries to retrieve text using the OCR (optical character recognition) mechanism. (This setting is highly recommended when working with CJK (Chinese, Japanese, Korean) languages.) - **First OCR then Windows API.** Instructs UFT to first try to retrieve text from the object using the OCR mechanism. If no text can be retrieved, then UFT uses its Windows API-based mechanism to retrieve text from the object. - **Use Only Windows API.** Instructs UFT to use only the Windows API-based mechanism (and not the OCR mechanism) to retrieve text from the object. - **Use Only OCR.** Instructs UFT to use only the OCR mechanism (and not the Windows API-based mechanism) to retrieve text from the object. (Required when working with Windows Vista; the only option available when working with Windows 7 and Windows Server 2008 R2.) For details, see "Notes and Guidelines" on the next page.
Single text block mode	Select this radio button if the text on the object is uniform in font, size, color, and background. For example:  Welcome ! The single text block mode instructs the OCR mechanism to focus on the area and treat it as a single text block. This is especially useful when trying to capture text on small objects or in a small text area.
Multiple text block mode	Select this radio button only if the text on the object comprises different fonts, font sizes, colors, and/or backgrounds. For example:  Welcome ! Welcome to HP QuickTest Professional, the advanced solution for functional test and regression test automation. This next-generation automated testing solution deploys the concept of keyword-driven testing to enhance test creation and maintenance. The multiple text block mode instructs the OCR mechanism to handle each text area in the object that has a different background font and size. The OCR mechanism decides where to divide the text blocks according to an internal algorithm.

UI Element	Description
Available languages	Lists all of the language dictionaries that the OCR mechanism can potentially use when retrieving text from the object. **To specify the language dictionaries used by the OCR mechanism:** Move a language to the **Supported languages** list box by selecting a language and clicking the right arrow button (**>**).
Supported languages	Lists the language dictionaries that the OCR mechanism uses when capturing text. The **Supported languages** list box can contain either: • One CJK (Chinese, Japanese, Korean) language. (**Note:** By default, English is also supported when capturing text in CJK languages.) • One or more non-CJK languages. **To remove a language dictionary from the Supported languages list:** Select the language and click the left arrow button (**<**).

Notes and Guidelines

- The Windows API text recognition mechanism is provided as is. If it does not work as you expect, change the option to use OCR. (Although OCR accuracy is usually very high, keep in mind that as with all OCR technologies, the OCR mechanism does not guarantee 100% accuracy.) For details, see "Guidelines for Text Recognition" on page 1409.

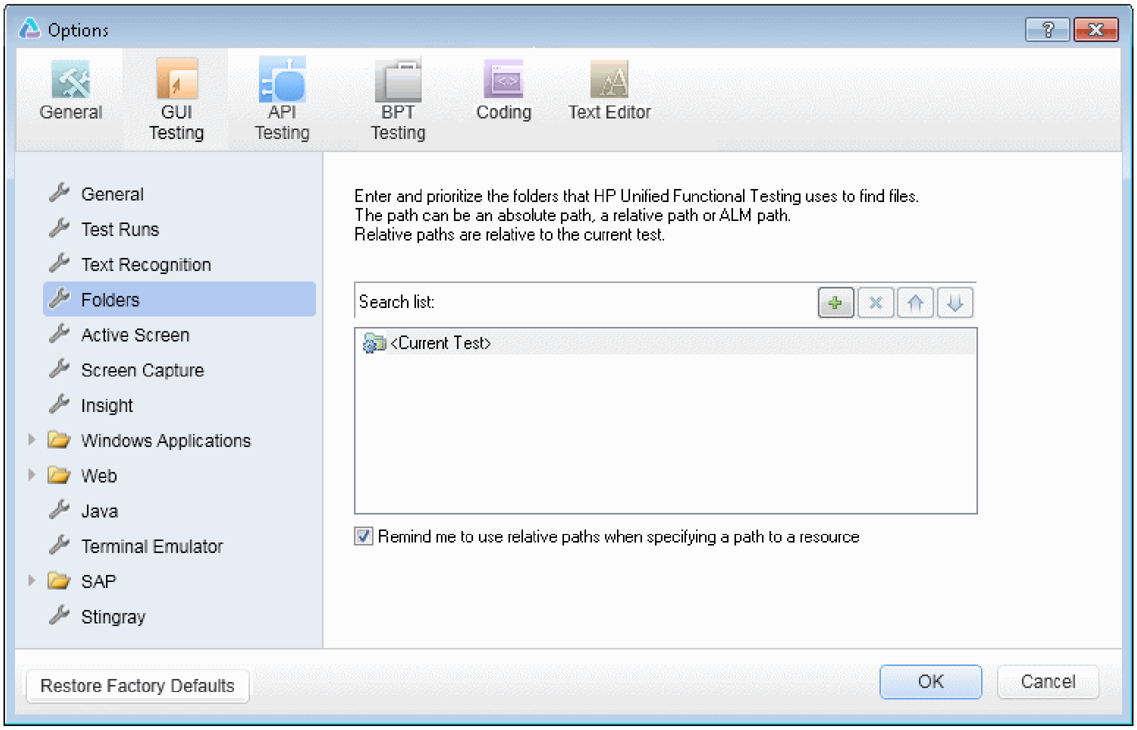
- **Guidelines for Windows Vista, Windows 7, Server 2008, Server 2008 R2, Windows 8, and Windows Server 2012 Users.** These operating systems do not support the Windows API text recognition mechanism. Therefore, only the **Use only OCR** option is available when UFT is installed on these operating systems.

Folders Pane (Options Dialog Box > GUI Testing Tab)

Relevant for: GUI tests and components

This pane enables you to enter the folders (search paths) in which UFT searches for tests, components, actions, or resource files that are specified as relative paths in dialog boxes and steps.

For example, suppose you add the folder in which all of your tests are stored to the folders list. If you later insert a copy of an action to a test, you only have to enter the name of the test containing the action you want to insert in the Insert Copy of Action dialog box. UFT searches for the test's path in the folders you specified in the Folders pane. For components and application areas, all files must be stored in an ALM path.



Enter and prioritize the folders that HP Unified Functional Testing uses to find files.
The path can be an absolute path, a relative path or ALM path.
Relative paths are relative to the current test.

Search list:

<Current Test>

☑ Remind me to use relative paths when specifying a path to a resource

To access	1. 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. 2. Select **Tools > Options > GUI Testing** tab > **Folders** node.

Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • The current test or component is listed in the **Search list** by default. It cannot be deleted. • If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, specify an absolute Quality Center or ALM path. For details, see "Relative Paths and ALM" on page 758. • You can use a **PathFinder.Locate** statement in your test to retrieve the complete path that UFT uses for a specified relative path based on the folders specified in the Folders pane. For details, see the *HP UFT Object Model Reference for GUI Testing*. • UFT searches for the specified test, component, action, or file in the order in which the folders are displayed in the search list. If the same file name exists in more than one folder, UFT uses the first instance it finds.
See also	• For details on relative or absolute paths, see "Relative Paths in UFT for GUI Testing" on page 130.

User interface elements are described below:

UI Element	Description
Search list	Indicates the folders in which UFT searches for tests, components, actions, or files. If you define folders here, you do not need to designate the full path of a test, component, action, or file in other dialog boxes or call statements. The order of the search paths in the list determines the order in which UFT searches for a specified action or file.
➕	Adds a new folder to the search list. **Tip:** • To add an ALM path when connected to ALM, click this button. UFT adds [ALM], and displays a browse button so that you can locate the ALM path. • When not connected to ALM, hold the SHIFT key and click this button. UFT adds [ALM], and you enter the path. You can also type the entire ALM path manually. If you do, you must add a space after [ALM]. For example: [ALM] Subject\Tests. • UFT searches ALM project folders only when you are connected to the corresponding ALM project.

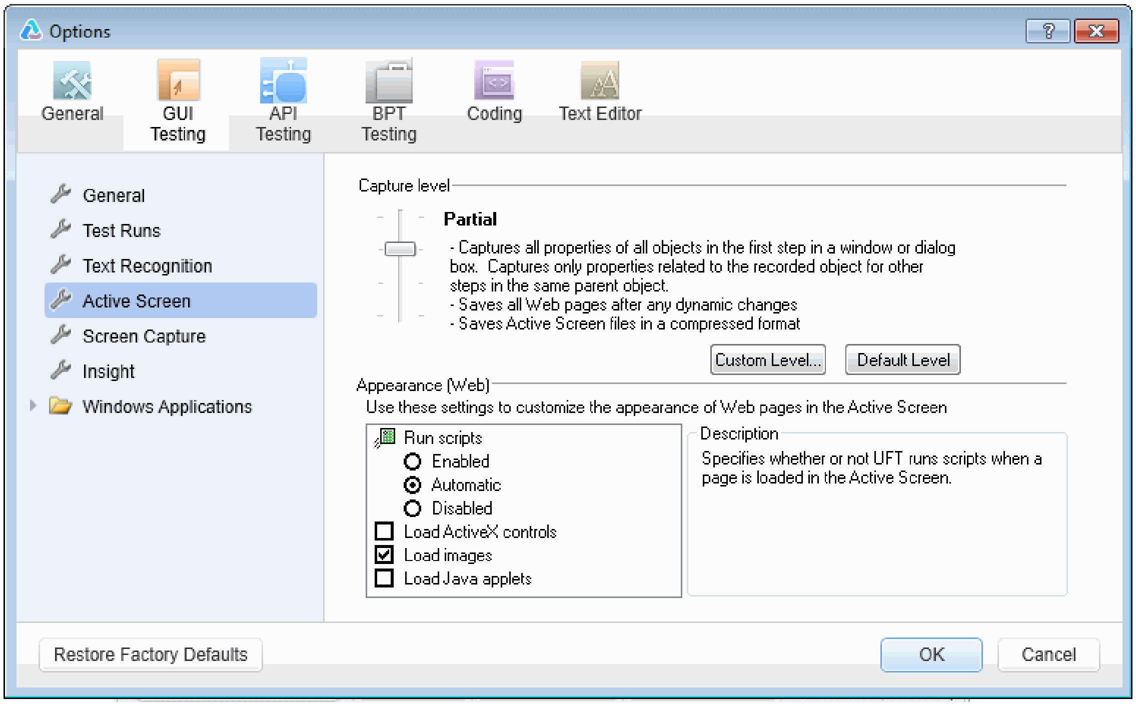
UI Element	Description
✗	Deletes the selected folder from the search list.
↑	Moves the selected folder up in the list.
↓	Moves the selected folder down in the list.
Remind me to use relative paths when specifying a path to a resource	When associating a resource, you can choose to be prompted to use a relative path. For details, see "Relative Paths in UFT for GUI Testing" on page 130. **Note:** When UFT is connected to a Quality Center 10.00 or ALM project, a reminder is displayed only if you select a path in the file system or in a Quality Center 9.2 project.

Active Screen Pane (Options Dialog Box > GUI Testing Tab)

Relevant for: GUI tests only

This pane enables you to specify which information UFT saves and displays in the Active Screen while recording and running tests.

The more information saved in the Active Screen, the easier it is to edit the test after it is recorded. However, more information saved in the Active Screen adds to the recording time and disk space required. This is especially critical with Windows-based add-ins, as they require more disk space to save Active Screen data.



Capture level

Partial

- Captures all properties of all objects in the first step in a window or dialog box. Captures only properties related to the recorded object for other steps in the same parent object.
- Saves all Web pages after any dynamic changes
- Saves Active Screen files in a compressed format

Appearance (Web)

Use these settings to customize the appearance of Web pages in the Active Screen

Run scripts
○ Enabled
● Automatic
○ Disabled
□ Load ActiveX controls
☑ Load images
□ Load Java applets

Description

Specifies whether or not UFT runs scripts when a page is loaded in the Active Screen.

To access	Select **Tools > Options > GUI Testing** tab > **Active Screen** node.
Important information	- The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. - When you are recording on an MDI (Multiple Document Interface) application, the Active Screen does not capture information for non-active child frames.
Related tasks	- "How to Modify the Active Screen Settings" on page 198 - "How to Use the Active Screen in Your Test" on page 197 - "How to Update Test Object Descriptions, Checkpoints, or Output Values, or Active Screen Captures" on page 1088
See also	- "Custom Active Screen Capture Settings Dialog Box" on page 551 - "Active Screen Overview" on page 194

The Active Screen pane contains the following key areas:

- "Capture Level Area" on the next page
- "Appearance (Web) Area" on page 550

Capture Level Area

UI Element	Description
Capture level	Specifies the objects for which UFT stores data in the Active Screen. Use the slider to select one of the following options: • **Complete.** Captures all properties of all objects in the application's active window/dialog box/Web page in the Active Screen of each step. This level saves Web pages after any dynamic changes and saves Active Screen files in a compressed format. • **Partial.** Captures all properties of all objects in the application's active window/dialog box/Web page in the Active Screen of the first step performed in an application's window, plus all properties of the recorded object in subsequent steps in the same window. This level saves Web pages after any dynamic changes and saves Active Screen files in a compressed format. • **Minimum (default).** Captures properties only for the recorded object and its parent in the Active Screen of each step. This level saves the original source HTML of all Web pages (prior to dynamic changes) and saves Active Screen files in a compressed format. • **None.** Disables capturing of Active Screen files for all applications and Web pages.
Custom Level	Enables you to specify custom Active Screen options. For details, see "Custom Active Screen Capture Settings Dialog Box" on page 551.
Default Level	Restores the capture level settings to the predefined default level (**Partial**).

Appearance (Web) Area

UI Element	Description
Appearance (Web)	Enables you to modify how UFT displays captured Web pages in the Active Screen. **Note:** • UFT loads ActiveX controls or Java applets to the Active Screen in view-only mode. You cannot perform operations or retrieve additional information on the loaded ActiveX or Java objects. To perform operations on these items from the Active Screen, you must load the relevant add-in and then record directly on the ActiveX or Java object. • ActiveX controls or Java applets that are loaded to the Active Screen may not work exactly as they do in the application. In some cases, this may cause unexpected behavior, depending on the implementation of the specific controls or applets that are loaded.
Run scripts	Specifies whether UFT runs scripts when a page is loaded in the Active Screen, according to one of the following options: • **Enabled.** Runs scripts whenever loading a page in the Active Screen. • **Automatic.** Runs scripts as needed, according to the page that is displayed. • **Disabled.** Prevents scripts from running when loading a page in the Active Screen. **Note:** This option refers only to scripts that run automatically when the page loaded. It does not enable you to activate scripts in the Active Screen by performing an operation on the screen. When working with the Web 2.0 ASPAjax Add-in, this option is disabled by default.
Load ActiveX controls	Instructs UFT to load ActiveX controls from your browser page to the Active Screen, so that for each step you can preview how the page is actually displayed in the application. If this option is cleared, a default ActiveX image is displayed in the Active Screen for all ActiveX control objects.
Load images	Instructs UFT to load images from your browser page to the Active Screen.

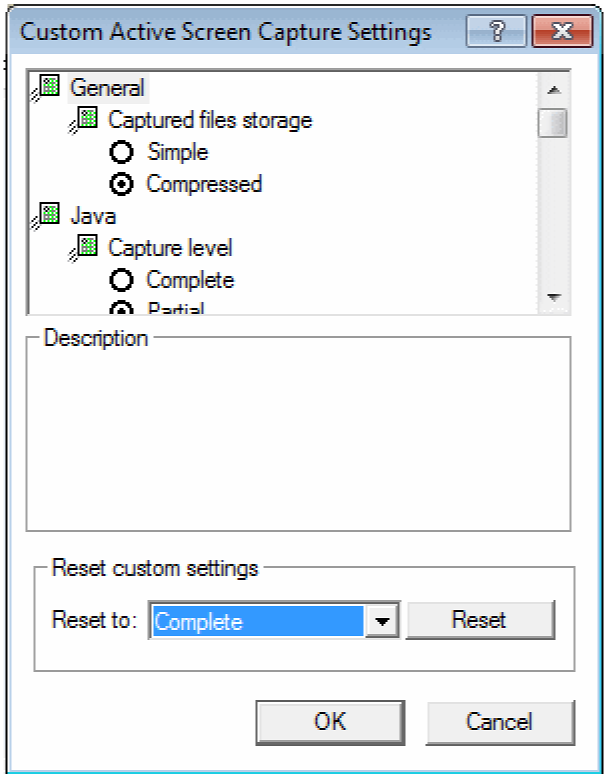
UI Element	Description
Load Java applets	Instructs UFT to load Java applets from your browser page to the Active Screen, so that for each step you can preview how the page is actually displayed in the application. If this option is cleared, a default Java image is displayed in the Active Screen for all Java applet objects.

Custom Active Screen Capture Settings Dialog Box

Relevant for: GUI tests only

This dialog box enables you to customize how UFT captures and saves Active Screen information.

When you apply custom Active Screen settings, you override the capture-level setting in the Active Screen pane with all of the settings in the Custom Active Screen Capture Settings dialog box.



Custom Active Screen Capture Settings

General
- Captured files storage
  - ○ Simple
  - ● Compressed

Java
- Capture level
  - ○ Complete
  - ● Partial

Reset to: Complete

To access	Select **Tools > Options > GUI Testing** tab > **Active Screen** node > **Custom Level.**

Important information	• The Custom Active Screen Capture Settings dialog box may also contain options applicable to any UFT add-ins installed on your computer. • The default settings in the Custom Active Screen Capture Settings dialog box do not reflect the selected capture-level setting in the Active Screen pane of the Options dialog box. If you want to customize only specific settings, use the **Reset to** option to ensure that all other settings are using the capture-level setting you prefer and then modify the specific settings you need.
Related tasks	• You can specify whether to save screen captures of the Active Screen using the **Save Active Screen files** option in the Save dialog box. See "Save <Resource>/Save <Document> As Dialog Box" on page 164. • You can use the **Update Run Mode** option to modify the amount of information saved in the Active Screen after you modify the Active Screen capture settings. See "Update Options Tab (Update Run Dialog Box)" on page 1107.

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

UI Element	Description
<settings box>	Enables you to select the specific setting options that determine how UFT captures and saves Active Screen information. The Settings box may also contain options applicable to UFT add-ins installed on your computer. For details, see: • "General" on the next page • "Capture Level - General" on the next page • "Capture Level - Java Applications or Applets" on the next page • "Capture Level - SAP GUI for Windows Applications" on page 554 • "Capture Level - Oracle Applications" on page 555 • "Capture Level - Windows Applications " on page 555 • "Capture Level - Terminal Emulator Applications" on page 556 • "Web" on page 556
Description	Provides a description of the option selected in the Settings box.
Reset custom settings	Enables you to reset the custom settings to one of the predefined levels provided with UFT (**Complete**, **Partial**, **Minimum**, or **None**) by choosing a level from the **Reset to** list and clicking the **Reset** button. For details on the available capture levels, see "Capture Level - General" on the next page.

General

By selecting a **Captured files storage** option, you can specify the type of compression UFT uses for storing captured Active Screen information:

- **Simple.** Instructs UFT to save Active Screen captures in standard uncompressed file formats (for example, .html and .png).
- **Compressed.** Instructs UFT to save Active Screen captures in a compressed (zipped) file format. Using this option saves disk space, but it may affect the time it takes to load images in the Active Screen. This is the default option.

Capture Level - General

The Custom Active Screen Capture Settings dialog box enables you to customize how UFT captures and saves Active Screen information.

Capture level options are available for Java applets or applications, SAP GUI for Windows applications, Oracle applications, Windows-based applications, and Terminal Emulator applications. The options available in the Custom Active Screen Capture Settings dialog box depend on the add-ins that are installed.

By selecting a **Capture level** option, you can specify which properties are captured for each object in an application when it is captured for the Active Screen.

Depending on your testing requirements, you can choose between different levels of Active Screen capture. However, you should take into consideration that the less information captured for the Active Screen, the better the performance.

For example, if you select the **Complete** capture level option, you can add checkpoints on every test object displayed in any Active Screen capture after recording, but it takes more time and use more disk space to record a single operation. Selecting **Partial** enables UFT to record faster and use less disk space, but there may be limitations on the operations you can perform from the Active Screen after recording.

The following sections describe the capture level options available for different environments.

Capture Level - Java Applications or Applets

The following **Capture level** options are available for Java applications or Java applets:

- **Complete.** Instructs UFT to save all identification properties of all objects in the application or applet's open window/dialog box in the Active Screen of each step.
- **Partial.** (Default) Instructs UFT to save all identification properties of all objects in the application or applet's open window/dialog box in the Active Screen of the first step performed in that window, plus all properties of the recorded object only, in subsequent steps in the same window.
- **Minimum.** Instructs UFT to save all identification properties for the recorded object plus all identification properties for the parent objects in the recording hierarchy.
- **None.** Disables capture of Active Screen files for Java applications or Java applets.

When the **Complete** or **Minimum** capture level is selected, the following setting in the Custom Active Screen Capture Settings dialog box is also relevant for Java applications or Java applets:

Disable capture of the following objects. Prevents UFT from capturing the data of steps performed on other objects for the selected test object types in the Active Screen. These objects are visible in the Active Screen as images only.

By default, **JavaObject** and **JavaMenu** are selected (meaning that identification properties are not captured for these objects).

> **Note:** If you record on a specific test object, its identification properties are captured even if the **Disable capture of the following objects** option is selected.

Capture Level - SAP GUI for Windows Applications

The following **Capture level** options are available for SAP GUI for Windows applications:

- **Complete.** Instructs UFT to save the property values of all objects in the application's open window/dialog box in the Active Screen of each step.

  This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box from the Active Screen of any step. However, it may result in longer recording times and require more disk space.

  > **Note:** The properties for inner objects of some container objects (such as table cells or tree nodes) are not captured in the Active Screen. Use the appropriate SAPGuiTable or SAPGuiTree methods to access information for these objects. For details, see the SAP GUI for Windows section of the *HP UFT Object Model Reference for GUI Testing*.

- **Partial.** (Default) Instructs UFT to save properties of the recorded object and of its parent in the Active Screen of each step.

  This option enables speedy recording and requires relatively little disk space. However, you can insert checkpoints and perform other operations only on the recorded object and on the window/dialog box itself. You cannot perform operations on the other objects displayed in the Active Screen.

- **None.** Disables the capture of Active Screen files for SAP GUI for Windows applications.

  This option allows extremely fast recording and requires only minimum disk space. However, you cannot perform post-recording test editing (such as inserting checkpoints, output values, and so forth) from the Active Screen.

  > **Notes:**
  >
  > - The property values of the objects in the Active Screen reflect the values at the time that the steps are added to your test (when information is sent to the SAP server). These values may potentially be different from the property values at the time that a particular

step is performed.

- The Active Screen captures only the visible part of the SAP GUI for Windows applications window at the time that the step is added to the test.

Capture Level - Oracle Applications

The following **Capture level** options are available for Oracle applications:

- **Complete.** Instructs UFT to save all identification properties of all objects in the application's open window/dialog box in the Active Screen of each step.
- **Partial.** (Default) Instructs UFT to save all identification properties of all objects in the application's open window/dialog box in the Active Screen of the first step performed in that window, plus all identification properties of the recorded object only, in subsequent steps in the same window.
- **Minimum.** Instructs UFT to save all identification properties for the recorded object plus all identification properties for the parent objects in the recording hierarchy.
- **None.** Disables capture of Active Screen files for Oracle applications.

Capture Level - Windows Applications

The following **Capture level** options are available for Windows applications.

- **Complete.** Instructs UFT to save all properties of all objects in the application's open window/dialog box in the Active Screen of each step.

  This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box, from the Active Screen for any step.

- **Partial.** (Default) Instructs UFT to save all properties of all objects in the application's open window/dialog box in the Active Screen of the first step performed in an application's window, plus all properties of the recorded object in subsequent steps in the same window.

  This option makes it possible for you to insert checkpoints and perform other operations on any object displayed in the Active Screen, while conserving recording time and disk space. Note that with this option the Active Screen information may not be fully updated for subsequent steps.

- **Minimum.** Instructs UFT to save properties only for the recorded object and its parent in the Active Screen of each step.

  This option enables speedy recording and requires relatively little disk space. However, you can insert checkpoints and perform other operations only on the recorded object and on the window/dialog box itself. You cannot perform operations on the other objects displayed in the Active Screen.

- **None.** Disables capture of Active Screen files for Windows applications.

  This option allows extremely fast recording and requires only a minimum of disk space. However, you cannot perform post-recording test editing from the Active Screen.

Capture Level - Terminal Emulator Applications

The following **Capture level** options are available for applications run on terminal emulators:

- **Complete.** Instructs UFT to save all properties of all objects in the application's open window/dialog box in the Active Screen of each step.

  This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box, from the Active Screen for any step.

- **None.** Disables capture of Active Screen files for Terminal Emulator applications.

Web

You can specify whether UFT captures Web pages for the Active Screen.

- **Disable Active Screen capture.** Disables the screen capture of all steps in the Active Screen.

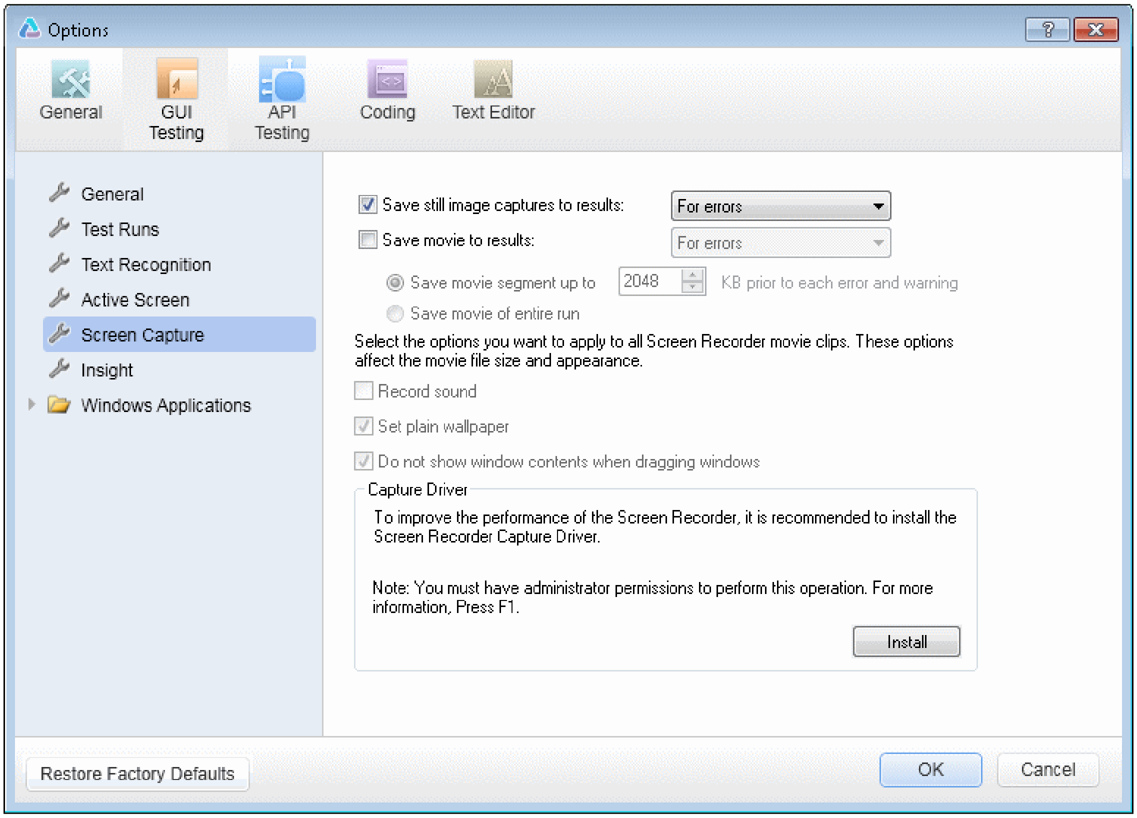
  If you do not select this option, you can also delete Active Screen information after you have finished editing your test by selecting **Save As**, and clearing the **Save Active Screen files** check box. For details, see "Save <Resource>/Save <Document> As Dialog Box" on page 164.

- **Capture original HTML source.** Captures the HTML source of Web pages as they appear originally, before any scripts are run. Deselecting this option instructs UFT to capture the HTML source of Web pages after any dynamic changes have been made to the HTML source (for example, by scripts running automatically when the page is loaded).

Screen Capture Pane (Options Dialog Box > GUI Testing Tab)

Relevant for GUI tests and components

This pane enables you to control how UFT captures images and movies of the application being tested.



☑ Save still image captures to results: For errors

□ Save movie to results: For errors

● Save movie segment up to 2048 KB prior to each error and warning

○ Save movie of entire run

Select the options you want to apply to all Screen Recorder movie clips. These options affect the movie file size and appearance.

□ Record sound

☑ Set plain wallpaper

☑ Do not show window contents when dragging windows

Capture Driver

To improve the performance of the Screen Recorder, it is recommended to install the Screen Recorder Capture Driver.

Note: You must have administrator permissions to perform this operation. For more information, Press F1.

To access	Select **Tools > Options > GUI Testing** tab > **Screen Capture** node.
Important information	- The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. - **Vista/Windows 7/Windows 8 users:** In addition to the options described below, if your Windows color scheme (or theme) is set to **Aero**, UFT automatically sets it to **Vista/Windows 7 Basic** while capturing movies of a run session to maximize performance. The color scheme is returned to its previous settings when the run session ends. For movies, see "Save movie to results" on page 559, below.
See also	How to Play a Screen Recorder Movie in the Micro Player (described in the *HP Run Results Viewer User Guide*)

This pane contains the following user interface elements:

UI Element	Description
Save still image captures to results	Instructs UFT when to capture still images of the application during the run session and save them in the run results. When images are available in the run results, UFT displays them in the Captured Data pane of the Run Results Viewer. • **Always.** Captures images for all steps in the run. • **For errors.** Captures images only for failed steps. This is the default setting. • **For errors and warnings.** Captures images only for steps that return a failed or warning status. **Note:** This setting also affects the availability of other information displayed in the Captured Data pane of the Run Results Viewer, such as: • XML checkpoint and output value result details • Bitmap checkpoint images (expected, actual, and, if relevant, difference)

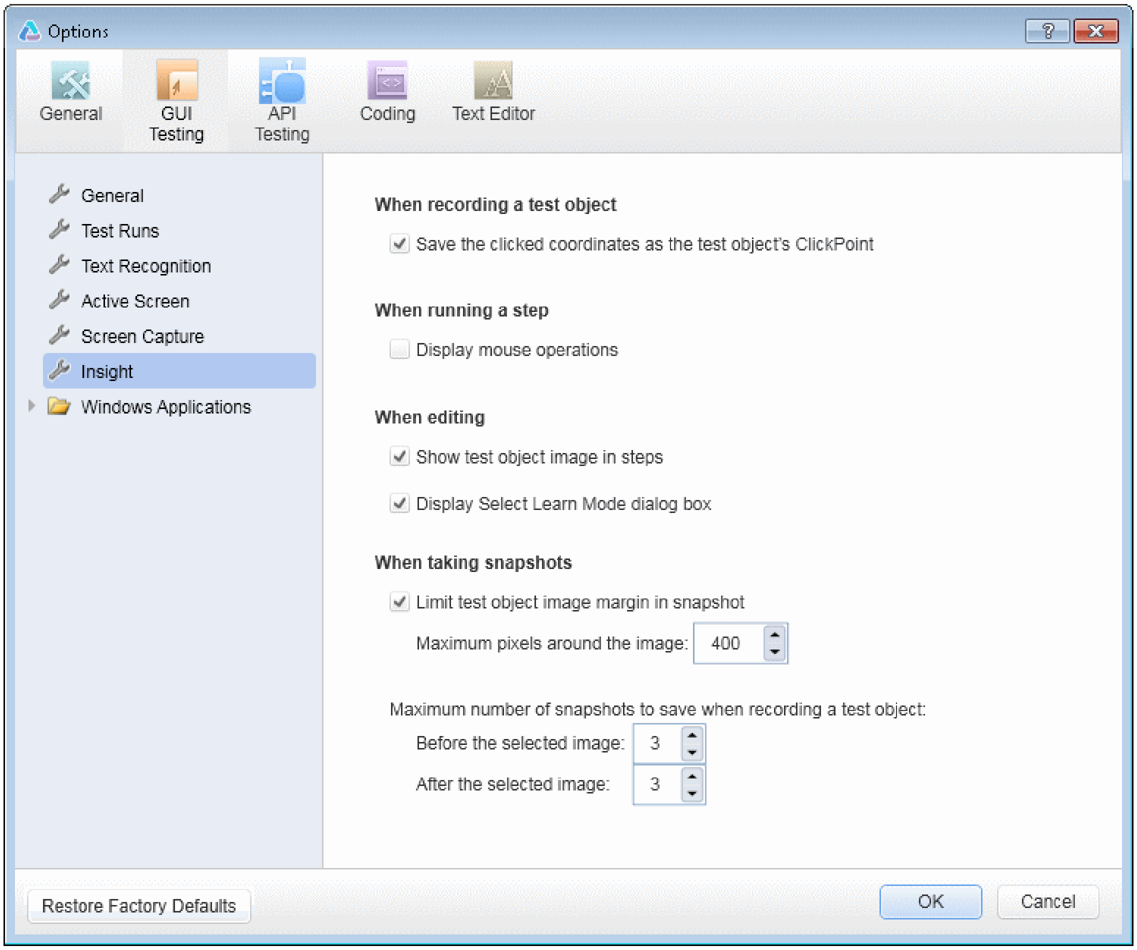
UI Element	Description
Save movie to results	Instructs UFT when to capture a movie of the application during the run session and save it in the run results. When movies are available in the run results, UFT displays them in the Screen Recorder tab in the Run Results Viewer. This option is disabled by default. Select the check box to enable this option and then select an option from the list: • **Always.** Captures a movie of all steps in the run. • **For errors.** Captures movies only for failed steps. • **For errors and warnings.** Captures movies only for steps that return a failed or warning status. **Important:** • **Recording only on primary monitor:** The Screen Recorder records a movie of the operations performed on your primary monitor. Therefore, if you are working with multiple monitors, make sure that your application is fully visible on your primary monitor when recording or running a test or component. • **Recording entire desktop.** The Screen Recorder saves a movie of your entire desktop. You can prevent the UFT window from partially obscuring your application while capturing the movie by minimizing UFT during the run session. For details on how to minimize UFT during run sessions, see "UFT Window Layout Customization in Different Modes" on page 183. • **Vista Users.** For best results when working with Microsoft Windows Vista, set the Window's display settings to the Classic Windows theme. • **Unicode Characters in Test and Component Names.** The HP Screen Recorder does not support Unicode. When you save a test or component using a name containing Unicode characters, and the run results contain a Screen Recorder movie, the movie is not saved.
Save movie segment up to __ KB prior to each error and warning	Instructs UFT to save movie segments for each error (or warning). Each segment contains the specified number of kilobytes of the movie prior to the failed (or warning) step. You can enter any value from 400 (0.4 MB) to 2097152 (2 GB). If more than one segment is captured for a run session, UFT stores a single movie comprised of all the relevant movie segments. **Note:** This option is enabled only when **For errors** or **For errors and warnings** is selected in the **Save movie to results** option.

UI Element	Description
Save movie of entire run	Instructs UFT to save a movie of the entire run if at least one error (or warning) occurs. **Note:** This option is enabled only when **For errors** or **For errors and warnings** is selected in the **Save movie to results** option.
Record sound	Instructs UFT to save sound with the movie of your application.
Set plain wallpaper	Sets the wallpaper of your desktop to a solid blue color for the duration of the run session.
Do not show window contents when dragging windows	Instructs Windows to display only the outline of a window, without its contents, whenever the window is dragged during the run session.
Install/Uninstall button	Installs or uninstalls the Screen Recorder Capture Driver. The Screen Recorder Capture Driver improves the performance of the Screen Recorder during movie recording. **Note:** The Screen Recorder Capture Driver cannot be installed or uninstalled when running UFT via a remote connection.

Insight Pane (Options Dialog Box > GUI Testing Tab)

Relevant for: GUI tests and components

This pane enables you to define options that customize how UFT handles Insight test objects when creating test object, and during record and run sessions.



When recording a test object

☑ Save the clicked coordinates as the test object's ClickPoint

When running a step

□ Display mouse operations

When editing

☑ Show test object image in steps

☑ Display Select Learn Mode dialog box

When taking snapshots

☑ Limit test object image margin in snapshot

Maximum pixels around the image: 400

Maximum number of snapshots to save when recording a test object:

Before the selected image: 3

After the selected image: 3

To access	Select **Tools > Options > GUI Testing** tab > **Insight** node.

Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • UFT captures additional snapshots of the application together with the Insight test objects that it creates. You can use these additional snapshots to adjust the test object images that identify the controls. The number of snapshots saved with a test object, as well as the test object image margin, affect the disk space required by your object repository. These settings can also affect UFT performance when recording and running Insight steps. ▪ Changes made in this pane affect only new snapshots and not previously saved ones. ▪ In the object repository, you can delete all of the snapshots stored with the Insight test objects after you finalize the test object images and verify that they enable correct object identification in all relevant scenarios.
Relevant tasks	• "How to Add an Insight Object to the Object Repository" on page 1214 • "How to Record a Test or Component Using Insight Recording" on page 860
See also	"Identifying Objects Using Insight" on page 1176

User interface elements are described below:

UI Element	Description
When recording a test object	
Save the clicked coordinates as the test object's ClickPoint	Instructs UFT to define the ClickPoint of each Insight test object added to the object repository during a recording session, according to the location that the user clicks. The coordinates of the first click on the object are used. • You can modify a test object's ClickPoint manually in the object repository. If this option is cleared, the ClickPoint for objects learned by recording is set to Center. • Modifying this option does not affect test objects previously added to the object repository. It affects only objects learned in subsequent recording sessions.
When running a step	
Display mouse operations	Instructs UFT to show mouse drag operations and clicks when running steps on Insight test objects.

UI Element	Description
When editing	
Show test object image in steps	Instructs UFT to display Insight test object images in steps in the Editor. If cleared, UFT displays the test object names instead.
Display Select Learn Mode dialog box	Instructs UFT to open the "Select Learn Mode Dialog Box" (described on page 1248) when you click the **Add Insight Object to Local** or **Add Insight Object** button to add an Insight object to an object repository.
When taking snapshots	
Limit test object image margin in snapshot	Instructs UFT to limit the snapshot size when creating a new Insight test object by including only a limited area of the screen around the control in the snapshot. Specify the margin size in the "Maximum pixels around the image" box. If cleared, UFT includes the whole screen in the snapshot. **Note:** Relevant when adding a new Insight test object (recording or learning).
Maximum pixels around the image	The maximum size of the area around the control to include in the snapshot. UFT includes up to the specified number of pixels on all sides of the control. **Maximum value**: 1000. **Note:** Available only when the "Limit test object image margin in snapshot" option is selected.

UI Element	Description
Maximum number of snapshots to save when recording a test object	Instructs UFT not to save more than the specified number of snapshots when recording a test object. Separately specify the number of snapshots to save **Before the selected image** and **After the selected image**. UFT selects the image that seems most appropriate to use for the object description, and then saves additional snapshots taken during the recording session, before or after the selected image, as available. The total number of snapshots saved may include up to the maximum number of snapshots you specified to save before and after the selected image, and one more snapshot for the test object image itself. **Maximum value:** 10 before and 10 after. **Note:** Relevant during a recording session.

API Testing Tab (Options Dialog Box)

Relevant for: API testing

The options in the API Testing tab of the Options dialog box enable you to customize your work environment when working with APIUFT tests or components. For example, you can define relative or absolute paths for saved files, repository locations, and auto values.

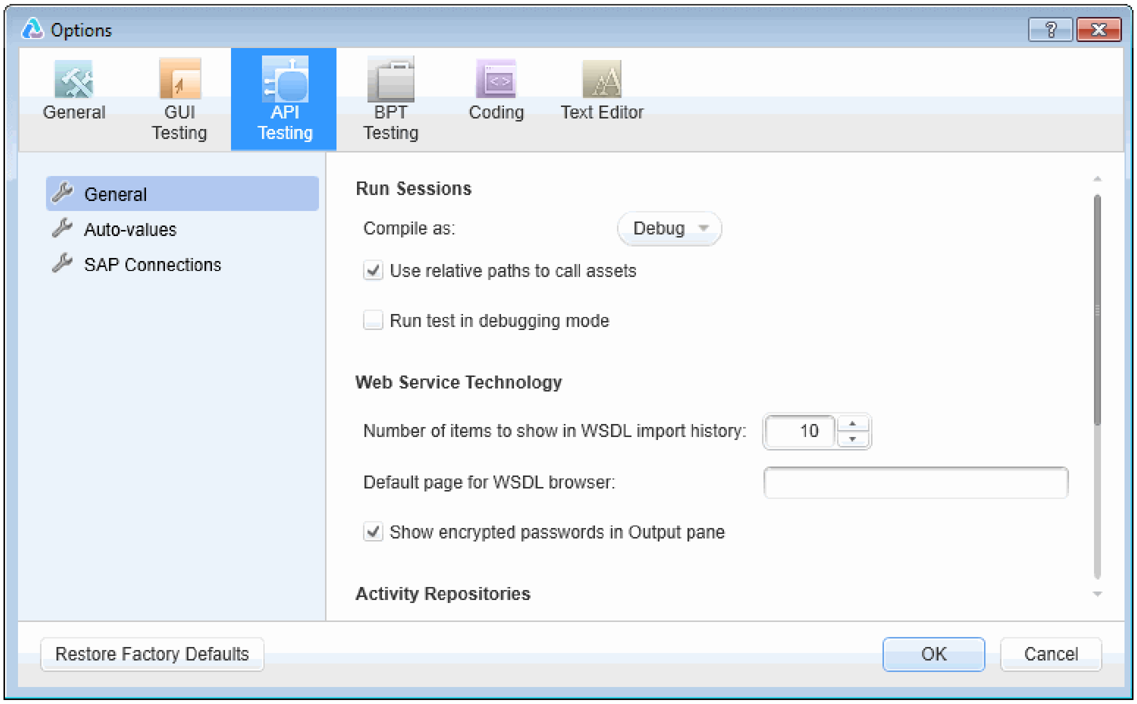
To access	Select **Tools > Options > API Testing** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "General Tab (Options Dialog Box)" on page 524 • "GUI Testing Tab (Options Dialog Box)" on page 535 • "Coding Tab (Options Dialog Box)" on page 572 • "Text Editor Tab (Options Dialog Box)" on page 574

This tab includes the following panes:

General Pane (Options Dialog Box > API Testing Tab)

Relevant for: API testing

This pane enables you to set the test execution mode, the default location for importing WSDL files, and the repository locations for activity sharing.



Run Sessions

Compile as: Debug

☑ Use relative paths to call assets

☐ Run test in debugging mode

Web Service Technology

Number of items to show in WSDL import history: 10

Default page for WSDL browser:

☑ Show encrypted passwords in Output pane

Activity Repositories

To access	Select **Tools > Options > API Testing** tab > **General** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	"Auto Values Pane (Options Dialog Box > API Testing Tab)" on page 567

User interface elements are described below:

UI Element	Description
Compile as	The mode for a run session. Options include **Debug** or **Release**. Select the **Release** option for faster run sessions. Run sessions performed with the **Debug** option run more slowly and generate additional information in the Output pane.
Use relative paths to call assets	Instructs UFT to use relative paths for all steps that require a path or file name.

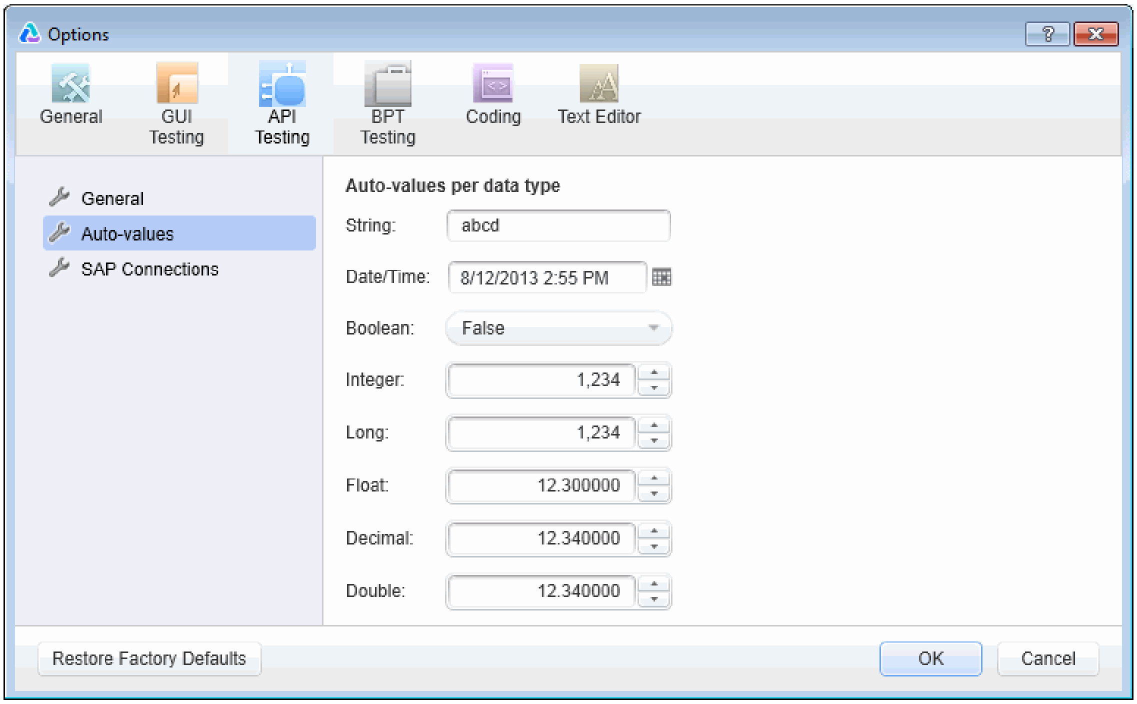
UI Element	Description
Run test in debugging mode	Runs a test with debugging capabilities active. By default, tests run without any of the debugging capabilities, enabling them to run faster. If you check this option, it activates the debugging capabilities during a test run, slowing the speed of a test run.
Number of items to show in WSDL import history	Specifies the maximum number of WSDL addresses to display in the address drop down list of the "Import/Update WSDL from URL or UDDI Dialog Box" (described on page 1766). **Default:** 10
Default page for WSDL browser	The default Web page that opens when you click the **Browse** button in the "Import/Update WSDL from URL or UDDI Dialog Box" (described on page 1766).
Shows encrypted passwords in the Output pane	Displays passwords used in a test or component in the information displayed in the Output pane in an unencrypted form.
Activity Repositories	Displays the locations for activity repositories: • **File system path.** The folder in the file system in which to store activities. • **ALM path.** The path in the ALM repository in which to store activities. **Tip:** • Click the **Browse** button to select a location. • To move an activity to a repository so that it can be used in subsequent tests, right-click the parent node and select **Move to**. **Note:** You must be connected to ALM to set the **ALM activities path** location (select **ALM > ALM Connection**).
Number of records to show in query preview	Sets the maximum number of strings for: • The Query Preview window (accessible by clicking the **Check SQL Statement** button in the "Set SQL Statement Page" (described on page 260). • A database source in the Data Pane **Default:** 10

UI Element	Description
Database connection timeout	The number of seconds to wait without a successful connection to a database before timing out the connection. **Default:** 30

Auto Values Pane (Options Dialog Box > API Testing Tab)

Relevant for: API testing

This pane enables you to provide default values for populating activity properties.



Auto-values per data type

String: abcd

Date/Time: 8/12/2013 2:55 PM

Boolean: False

Integer: 1,234

Long: 1,234

Float: 12.300000

Decimal: 12.340000

Double: 12.340000

To access	Select **Tools > Options > API Testing tab** > **Auto-values** node.
Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • To assign an auto value to a property, select **Set Auto-value** from the right-click menu in the **Value** column of the Properties pane. • To set values for an array of properties, select the parent node.
See also	"General Tab (Options Dialog Box)" on page 524

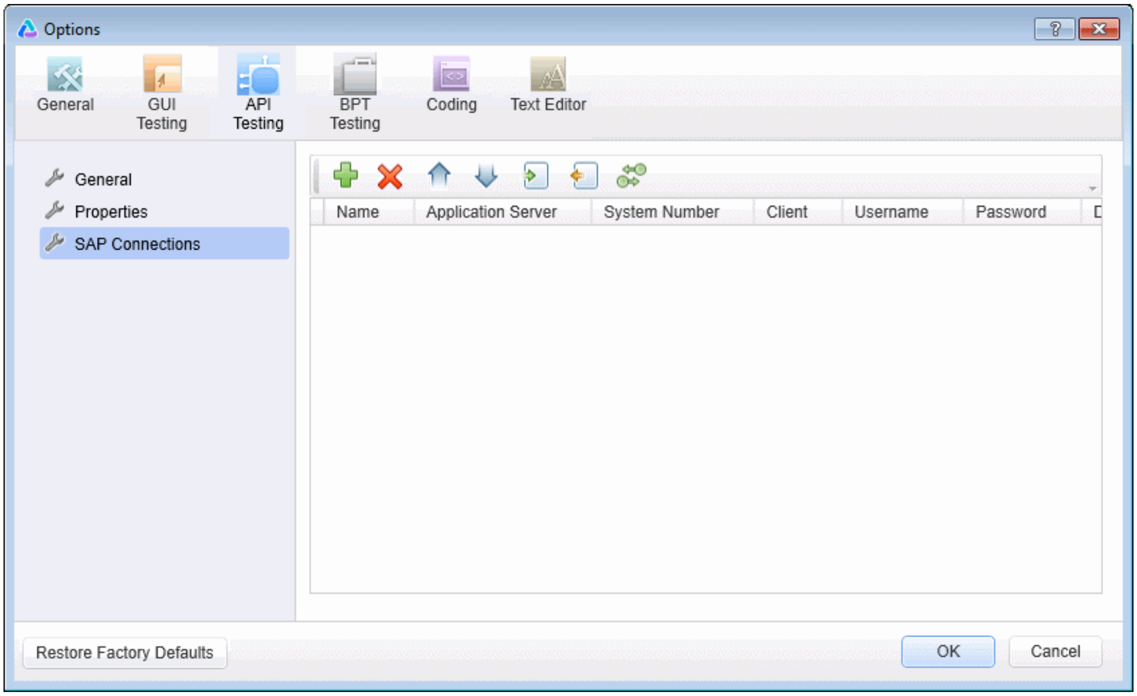
User interface elements are described below:

UI Element	Description
String	The default value to use for a property defined as a **string**. **Default:** abcd.
Date/time	The default value to use for a property requiring a time and date. **Default:** long notation of the current **time and date**.
Boolean	The default value to use for a property defined as **boolean**: True or False. **Default:** False.
Integer	The default value to use for a property defined as **integer**. **Default:** 1,234.
Long	The default value to use for a property defined as **Long**. **Default:** 1,234.
Float	The default value to use for a property defined as **Float**. **Default:** 12.300000.
Decimal	The default value to use for a property defined as **decimal**. **Default:** 12.340000.
Double	The default value to use for a property defined as **double**. **Default:** 12.340000.

SAP Connections Pane (Options Dialog Box > API Testing Tab)

Relevant for: API testing

This pane enables you to manage SAP connections. You can add and remove connections, add authentication information, and check the connection.



To access	Select **Tools > Options > API Testing tab > SAP Connections** node.
Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • To define a connection, you must have the 32-bit version of SAP .NET Connector installed on your machine. The installation is available on the SAP Help portal, at http://help.sap.com/saphelp_NW04/helpdata/en/e9/23c80d66d08c4c8c044a3ea11ca90f/content.htm.
Relevant tasks	"How to Create an SAP API Test Step" on page 1638
See also	"SAP Activities" on page 1737

User interface elements are described below:

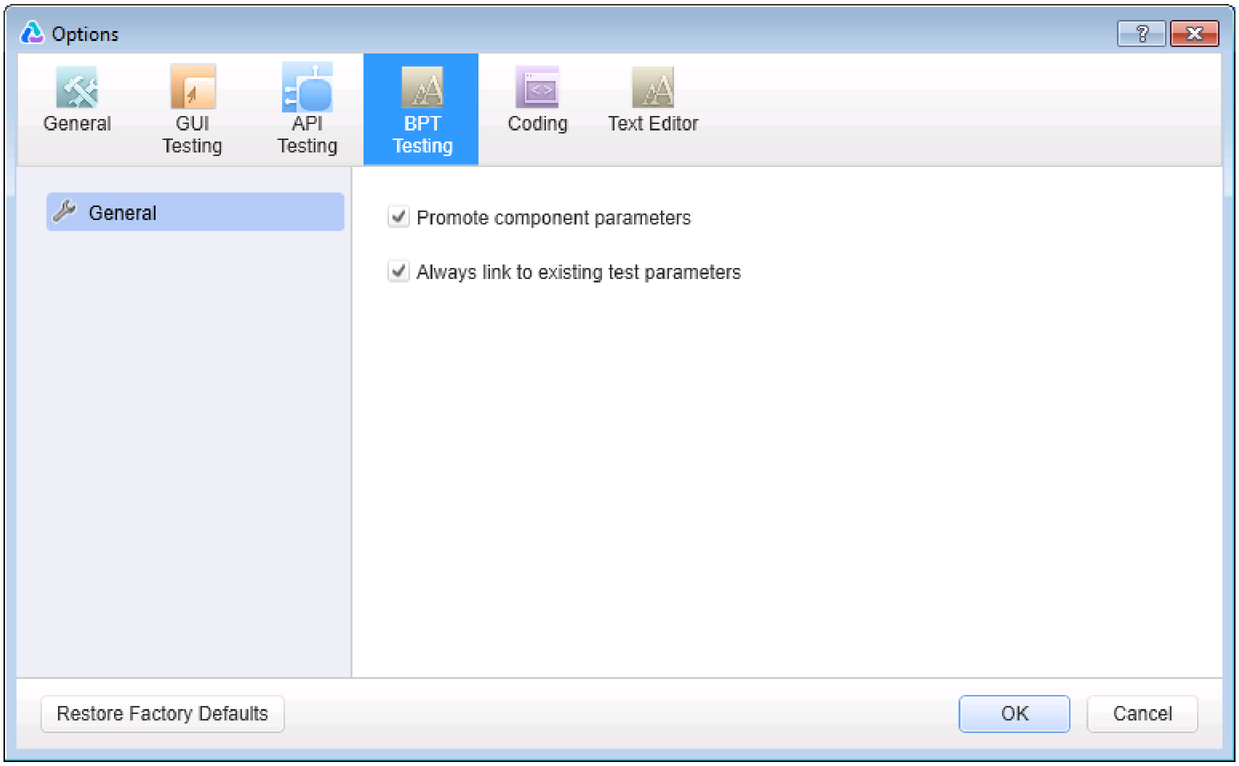
UI Element	Description
✚	**Add Connection.** Adds a new editable entry to the list of connections.
✖	**Delete Connection.** Deletes the selected connection.
⇧	**Move up.** Moves the selection up on the connection list.
⇩	**Move down.** Moves the selection down in the connection list.

UI Element	Description
→	**Import.** Allows you to import a saplogon and connection file (.ini or .xml) with the connection settings information.
←	**Export.** Allows you to export the connection settings to an XML file.
⇄	**Ping Connection.** Tests the selected connection and inserts the date and time in the **Last Pinged** column.
<SAP connection list>	Displays information about the current connection: • **Name.** The connection's name. • **Application Server.** The URL or IP of the application server. • **System Number.** The value of the SAP system number. • **Client.** The type of client. • **Username/Password.** The authentication information. • **Description.** A description of the connection. • **Last Pinged.** The date and time that a ping request was issued to the server.

BPT Testing Tab (Options Dialog Box)

Relevant for: business process tests and flows

This tab enables you to set options for parameter use of a BPT test and its components in UFT.



☑ Promote component parameters

☑ Always link to existing test parameters

To access	Select **Tools > Options > BPT Testing** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Element	Description
Promote component parameters	Determines whether UFT automatically promotes parameters to the flow or test level (when adding a component to a flow or test) or promotes flow parameters to the test level (when adding a flow to a test).
Always link to existing test parameters	Determines whether UFT uses an existing test parameter during promotion or creates an additional test parameter, when a parameter with the same name already exists in the test or flow. If selected, the Test/Flow Parameter Name suffix is removed, because the parameter may have been promoted from multiple components.

Coding Tab (Options Dialog Box)

Relevant for: GUI testing and API testing

This tab enables you to set options that facilitate designing code.

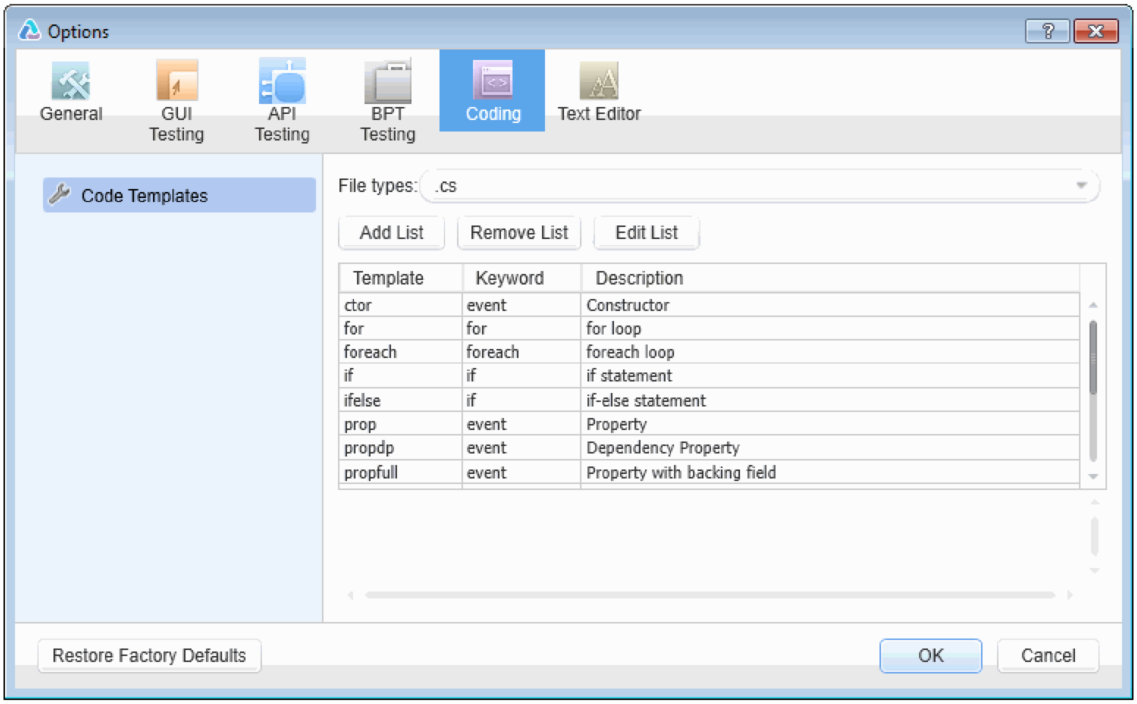
To access	Select **Tools > Options > Coding** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "General Tab (Options Dialog Box)" on page 524 • "GUI Testing Tab (Options Dialog Box)" on page 535 • "API Testing Tab (Options Dialog Box)" on page 564 • "Text Editor Tab (Options Dialog Box)" on page 574

This tab includes the following pane:

Code Templates Pane (Options Dialog Box > Coding Tab)

Relevant for: GUI testing and API testing

This pane enables you to create and edit templates of pre-designed code snippets or blocks of text used for automatic code completion. You can create different lists of templates for different file types.



File types: .cs

Template	Keyword	Description
ctor	event	Constructor
for	for	for loop
foreach	foreach	foreach loop
if	if	if statement
ifelse	if	if-else statement
prop	event	Property
propdp	event	Dependency Property
propfull	event	Property with backing field

To access	Select **Tools > Options > Coding** tab > **Code Templates** node.
Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • You can edit the names, keywords, descriptions, and code snippets of any template defined in this pane. • UFT provides default file types and snippets. You can add additional lists of file types and snippets as needed.
Relevant tasks	"How to Use Code Snippets and Templates" on page 322
See also	**For GUI testing and API testing:**"Automatic Code Completion" on page 316 **For GUI testing only:** • "Comments, Control-Flow, and Other VBScript Statements " on page 1006 • "Basic VBScript Syntax" on page 1013

User interface elements are described below:

UI Element	Description
File types	The file types for which the displayed list of templates is used. Select an item from the drop-down list to display the templates associated with the selected file extensions. Default options include: • .vb files . • .cs files (used for API tests) • .mts, .qfl, or .vbs files (used for GUI tests)
Add List	Adds a new list of templates and the file types for which to use them. A new item is added to the **File types** drop-down list.
Remove List	Removes the currently displayed list of templates and the associated **File types** item. **Caution:** Deleting a list is irreversible.
Edit List	Enables you to edit the file extensions associated with the displayed list of templates.
Template	The name of the template.
Keyword	The keywords that trigger the use of the template when invoking code completion in files of the selected types.
Description	A string that describes the code snippet.
<code snippet>	The code that is inserted into your document. The snippet can include text that is inserted as is, as well as variables (prefaced by a dollar sign $) that are replaced with actual values when they are inserted into the document. **Tip:** To edit the code snippet, modify the text in this area and click another template row or **OK**.

Text Editor Tab (Options Dialog Box)

Relevant for: GUI testing and API testing

This tab enables you to set options for editing text and code files.

To access	Select **Tools > Options > Text Editor** tab.

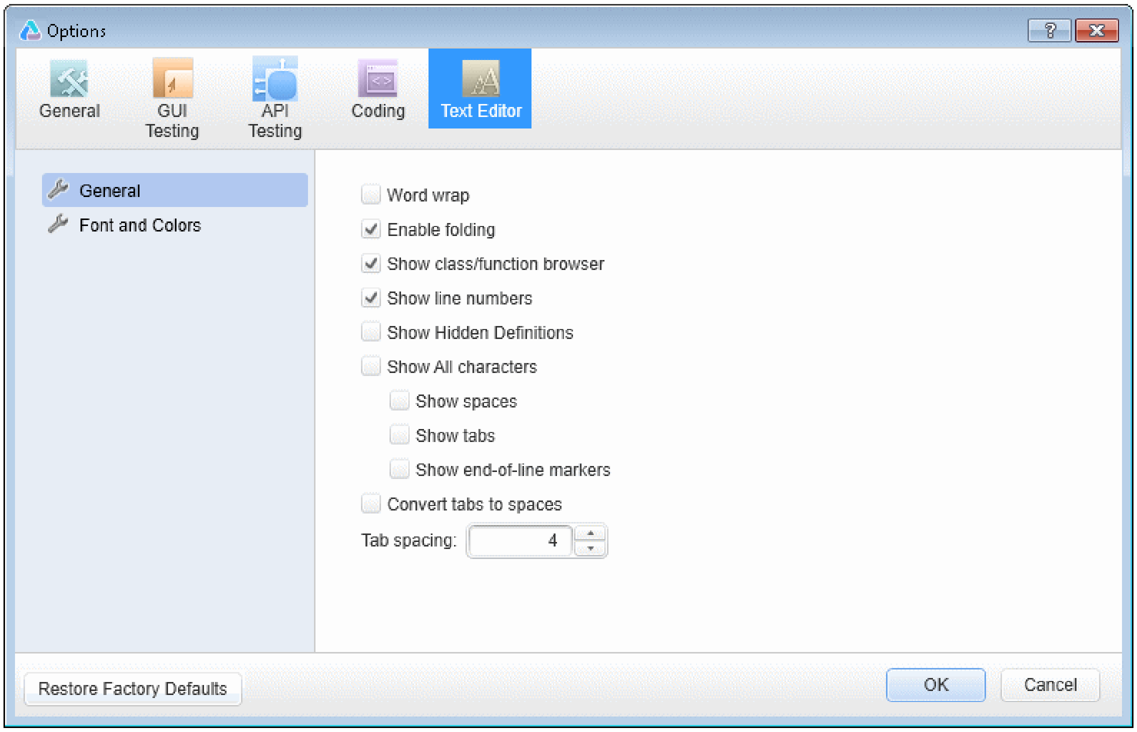
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	- "General Tab (Options Dialog Box)" on page 524 - "GUI Testing Tab (Options Dialog Box)" on page 535 - "API Testing Tab (Options Dialog Box)" on page 564 - "Coding Tab (Options Dialog Box)" on page 572

This tab includes the following panes:

General Pane (Options Dialog Box > Text Editor Tab)

Relevant for: GUI testing and API testing

This pane enables you to set general preferences for editing code and text files.



□ Word wrap
☑ Enable folding
☑ Show class/function browser
☑ Show line numbers
□ Show Hidden Definitions
□ Show All characters
□ Show spaces
□ Show tabs
□ Show end-of-line markers
□ Convert tabs to spaces
Tab spacing: 4

To access	Select **Tools > Options > Text Editor** tab > **General** node.

Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	- "Editing Text and Code Documents" on page 305 - "Editor User Interface" on page 334

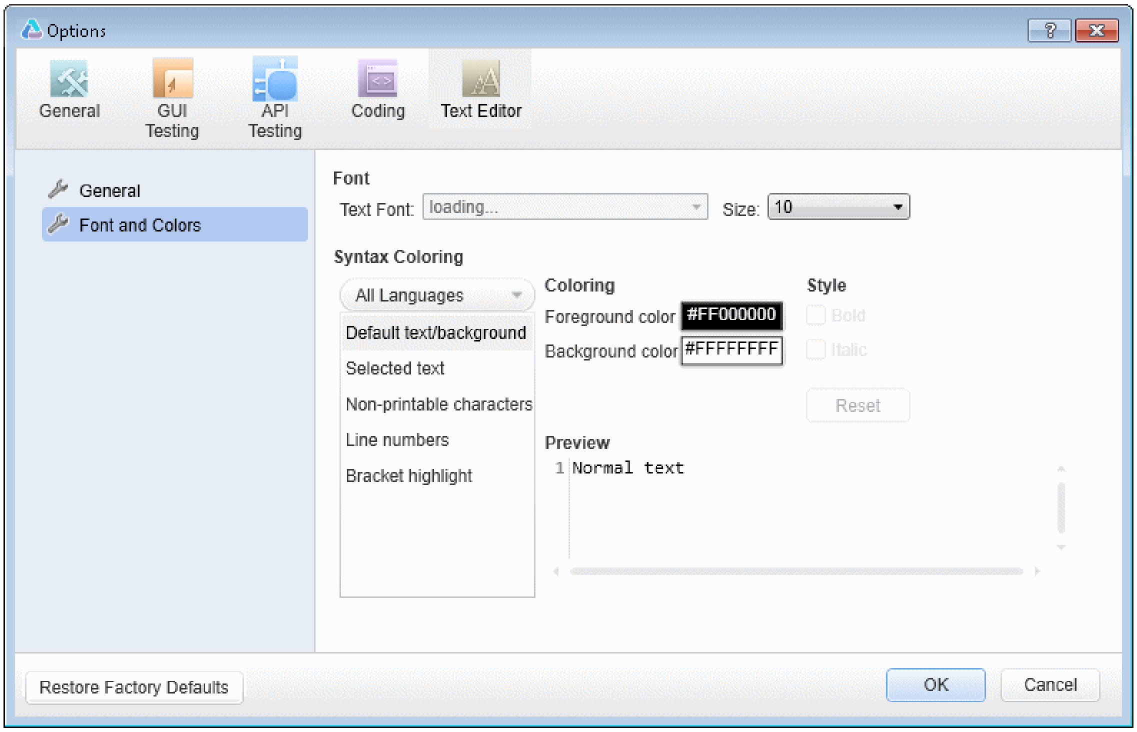
User interface elements are described below:

UI Element	Description
Word wrap	Wraps lines of code to continue onto subsequent lines without the insertion of a line break character. If disabled, the line of code continues indefinitely on a single line until you manually stop it by pressing the **ENTER** key or entering a line break character.
Enable folding	Enables sections of code that are grouped together to be expanded and collapsed.
Show class/function browser	Shows or hides the drop-down lists of classes and functions. These lists are located at the top of the Editor. Using the browsers, you can navigate to different classes and functions within your document.
Show line numbers	Shows or hides the line number to the left of each line of code.
Show hidden definitions	Shows or hides text and function definitions defined as hidden.
Show All characters	Displays all **SPACE**, **TAB**, and **End of Line** characters. You can also select to display only some of these characters by selecting or clearing the relevant check boxes.
Convert tabs to spaces	Changes **TAB** keystrokes in your code to the number of **SPACE** characters defined in the **Tab spacing** option.
Tab spacing	The number of characters inserted into a line of code for a **TAB** character. The number of characters can range from 1 to 16. **Default:** 4

Fonts and Colors Pane (Options Dialog Box > Text Editor Tab)

Relevant for: GUI testing and API testing

This pane enables you to specify color preferences for the text you are editing. You can set unique coloring rules for each code element.



Text Font: loading...

Size: 10

Foreground color #FF000000

Background color #FFFFFFFF

```
1 Normal text
```

To access	Select **Tools > Options > Text Editor** tab > **Fonts and Colors** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. The options in this pane apply only to documents in the Editor. For details on the GUI testing Keyword View formatting options, see "Fonts and Colors Tab (Keyword View Options Dialog Box)" on page 955.
See also	• "Editing Text and Code Documents" on page 305 • "Editor User Interface" on page 334

User interface elements are described below:

UI Element	Description
Font	The font name and size to use for all elements. The Default is Consolas, 10 points. **Note:** When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your document may not be correctly displayed in the document window. However, the test or component still runs in the same way, regardless of the font you choose. If you are working in an environment that is not Unicode-compatible, you may prefer to choose a fixed-width font, such as Courier, to ensure better character alignment.
Syntax Coloring	The language or languages whose appearance you want to customize. Available options: • **All languages** • **C#** • **VBScript** • **XML** The box below your selection displays all of the elements for which you can configure color and style settings. You can modify the **Foreground** and **Background** colors for a selected element. You can also modify the **Style** (**Bold** or **Italic**), if enabled.
Preview	An example of the text element according to your selections above.
Reset	Resets all colors and styles to default settings.

Troubleshooting and Limitations - Options Dialog

Relevant for: GUI tests and components

This section describes troubleshooting and limitations for the Options Dialog.

- UFT cannot resolve ALM folders paths specified in the Folders pane of the Options dialog box, if any of the folder names contain a forward slash.

  Workaround: Remove the slash in the folder name in ALM or specify paths using this folder using absolute paths (instead of relative paths that reference the Folders pane).

Chapter 20: Settings for GUI Tests, GUI Business Components, and Application Areas

Relevant for: GUI tests and components

This chapter includes:

Concepts

Settings Overview

Relevant for: GUI tests and components

You can use the Test or Business Component Settings dialog box or the Additional Settings pane in an application area to set testing options that affect how UFT works with a specific test or component. Most of the settings for components are defined in application areas and their settings.

Different panes and settings are available, depending on the document (test, component, or application area) that is in focus when you open the Settings dialog box or the Additional Settings pane.

> **Note:** You can also set testing options that affect all tests and components. For details, see "UFT Global Options" on page 522.

GUI Test Settings

Use the Test Settings dialog box to set testing options that affect how UFT works with a specific test. For example, you can instruct UFT to run a parameterized test for only certain lines in the data table. The individual testing options that you specify are saved when you save the test.

GUI Business Component Settings

Business components are automatically linked to the settings defined in their associated application area. For details, see "Application Area Settings " below. The Business Component Settings dialog box displays most of these settings in read-only format.

You can also define some additional component-specific settings, such as input and output parameters, component snapshots, and the component status, in the Business Component Settings dialog box.

Application Area Settings

You define the settings and resources needed for a business component in its associated application area.

The following application area settings are used only if the application area is associated with the first business component in a business process test:

- Associated Add-ins
- Local System Monitor
- Log Tracking

If you need a particular add-in or you want the local system monitor or log tracking options to be active for your business process test, you must make sure that the required settings are applied in the application area for the first component in that test. If these settings are active for application areas associated with later components in the test, they are ignored.

For details on working with application areas, see "Application Areas" on page 2095.

The Settings Dialog Box Panes

The table below lists the panes contained in the Settings dialog box. Some of these panes are available in the application area's Additional Settings pane.

Node	Options
Properties	The properties of the test or component, for example, its description and associated add-ins. You can also set the status of a component. For details, see "Properties Pane (Test/Business Component Settings Dialog Box)" on page 590.
Snapshot (components only)	Options for capturing or loading a snapshot image to be saved with the component for display in ALM. For details, see "Snapshot Pane (Business Component Settings Dialog Box)" on page 596.
Run (tests only)	The run session preferences for your test. For details, see "Run Pane (Test Settings Dialog Box)" on page 601.
Applications (components only)	The Windows-based applications on which the component can record and run. For details, see "Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 597. **Note:** • For components, define in the application area's Additional Settings pane. • Read-only in the Business Component Settings dialog box.
Resources	The resources associated with your test or component, such as function libraries, shared object repositories, and data tables. For details, see "Resources Pane (Test/Business Component Settings Dialog Box) " on page 603. **Note:** • For components, define in the application area. • Read-only in the Business Component Settings dialog box.

Node	Options
Environment (tests and scripted components only)	Options for viewing existing built-in and user-defined environment variables, adding, modifying and saving user-defined environment variables, and selecting the active external environment variables file. For details, see "Environment Pane (Test Settings Dialog Box)" on page 608.
Recovery	Options for setting how UFT recovers from unexpected events and errors that occur in your testing environment during a run session. For details, see "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 613. **Note:** • For components, define in the application area's Additional Settings pane. • Read-only in the Business Component Settings dialog box.
Log Tracking	Options for activating and setting run-time preferences for tracking log messages generated by the log framework that monitors events occurring in your application. For details, see "Log Tracking Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 617.
Local System Monitor	Options for activating and setting preferences for tracking system counters during a run session. For details, see "Local System Monitor Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 623. **Note:** For components, define in the application area's Additional Settings pane.

The Settings dialog box may also contain additional panes corresponding to any UFT add-ins that are installed or loaded. For details on add-ins, see the relevant section in the *HP Unified Functional Testing Add-ins Guide*.

Add-in Associations in a GUI Test

Relevant for: GUI tests only

For components, see "Add-in Associations in a GUI Component " on the next page.

When you open UFT, you select the add-ins to load from the Add-in Manager dialog box. You can create and edit tests that work with any environment for which the necessary add-in is loaded.

When you create a new test, the add-ins that are currently loaded are automatically associated with your test.

Choosing to associate an add-in with your test instructs UFT to check that the associated add-in is loaded each time you open that test. For details, see "Add-in Manager Dialog Box" on page 118.

When you open a test, UFT notifies you if an associated add-in is not currently loaded, or if you have loaded add-ins that are not currently associated with your test. This process ensures that your run session will not fail due to unloaded add-ins and reminds you to add required add-ins to the associated add-ins list if you plan to use them with the currently open test. For details on loading and working with add-ins, see the *HP Unified Functional Testing Add-ins Guide*.

ALM uses the associated add-ins list to determine which add-ins to load when it opens UFT to run or view a test. For details on working with ALM, see "ALM Integration" on page 707.

Add-in Associations in a GUI Component

Relevant for: GUI components only

For tests, see "Add-in Associations in a GUI Test" on the previous page.

When you open UFT, you can select the add-ins to load from the Add-in Manager dialog box. You can record on any environment for which the necessary add-in is loaded.

Choosing to associate an add-in with an application area instructs UFT to check that the associated add-in is loaded each time you open a component that is associated with that application area. When you create a new component, its associated add-ins are those defined in the component's application area. For details, see "Add-in Manager Dialog Box" on page 118.

When you open a component, UFT notifies you if an associated add-in is not currently loaded, or if you have loaded add-ins that are not currently associated with your component (via its application area). This process reminds you to add the required add-ins to the associated add-ins list if you plan to use them with the currently open component, thereby helping you to ensure that your run session will not fail due to unloaded add-ins.

When a business process test is opened in ALM, the UFT add-ins that are associated with the first component in the business process test are loaded automatically. Add-ins associated with other components in the business process test are not loaded. Therefore, it is important to ensure that all required UFT add-ins are associated with the application area of the first component in the business process test.

Understanding GUI Component Statuses

Relevant for: GUI components only

You can assign statuses to business components in UFT or in ALM. You can specify business component statuses manually, or in certain cases, ALM may automatically assign them. For example, you can use a **Ready** status to indicate that a business component is ready to be run in a business process test, or an **Error** status may be automatically assigned to a component that has errors that prevent it being successfully run in a business process test.

Knowing the status of a business component is important because it affects the status of any business process tests of which it is a part. In general, the component with the most severe status determines the status of the entire business process test. For example, a component with an **Error** status causes every business process test of which it is a part to have an **Error** status.

A component can be assigned one of the following statuses:

- **Error.** The component contains errors that need to be fixed. For example, this may occur due to a change in the application. When a business process test contains a component with this status, the status of the entire business process test is also **Error**.
- **Maintenance.** The component is currently being developed and tested and is not yet ready to run, or it was previously implemented and is now being modified to adapt it for changes that have been made in the application.
- **Ready.** The component is fully implemented and ready to be run. It answers the requirements specified for it and has been tested according to the criteria defined for your specific system.
- **Under Development.** The component is currently under development. This status is automatically assigned to:
  - New components created in the Business Components module of ALM with BPT support.
  - Component requests dragged into the component tree in ALM with BPT support.
- **Not Implemented.** The component has been requested in the Test Plan module of ALM. The status changes automatically from **Not Implemented** to **Under Development** when the request is moved from the Component Requests folder in the component tree in the Business Components module.

For details on setting component statuses, see "Properties Pane (Test/Business Component Settings Dialog Box)" on page 590.

Local System Monitor

Relevant for: GUI tests and components

Local system monitor tracking enables you to track application performance counters during a run session. These counters enable you to monitor the resources used by your application.

The system counters that can be monitored are the process counters that are accessible through the performance console (select **Start > Run** > and then enter Perfmon). For details on these process counters, see the performance console's Help.

You can also define limits for the counters. If the specified counters exceed these limits, the run session will fail. The results of the system counters are viewed in the Run Results Viewer. For details, see Run Results Viewer (described in the *HP Run Results Viewer User Guide*).

You can use the Local System Monitor pane of the Test Settings dialog box (for tests) or application area's Additional Settings pane (for components) to activate and set preferences for tracking system counters during a run session. For details, see "Local System Monitor Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 623.

When Running a Business Process Test

- Local system monitoring is performed only if the application area associated with the first business component in the business process test is configured to do so.

- Local system monitoring is performed only on the application defined in the Local System Monitor pane of the application area additional settings for the first component in the business process test.

  This means that if a component running later in the test is associated with a different application area, local system monitor settings from that application area are ignored and local system monitoring is not performed on that application's process.

- Local system monitoring can only monitor the status of one instance of the application. If more than one instance of the application is running, UFT monitors only the instance that was opened first.

Log Tracking

Relevant for: GUI tests and components

If the Windows-based application you are testing uses a supported Java or .NET log framework that includes a UDP appender, you can enable UFT to receive log messages from that framework and send them to the run results.

You do this by configuring your application's log configuration file. You can configure this file using UFT, or you can configure the file manually. To configure this file using UFT, you define the settings in the Log Tracking pane in the Settings dialog box described on page 586. For details on configuring the log configuration file manually, see "How to Manually Configure Log Tracking Settings" on the next page.

When you configure your application's log configuration file, you specify the minimum log message level that you want UFT to receive, the log message source and port, and so on. During a run session, UFT receives the relevant log messages that are generated for your application and sends them to the run results. In the Run Results Viewer, these log messages are displayed in the Log Tracking Results pane. In the Run Results Viewer, log messages are associated with steps according to their timestamp. For details, see the section on the Run Results Viewer Log Tracking Pane (described in the *HP Run Results Viewer User Guide*).

Analyzing the log messages that were generated as a result of a particular step can help you pinpoint the causes of unexpected behavior in your application, such as application crashes during automated runs.

For more details, see:

- "How to Manually Configure Log Tracking Settings" on the next page
- "Log Tracking Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 617

Tasks

How to Manually Configure Log Tracking Settings

Relevant for: GUI tests and components

You can specify your log tracking and collection preferences manually. This is useful, for example, if your test or component is validating more than one application, and you need to configure a file for each application being tested.

> **Note:** If you want to instruct UFT to automatically configure your application's log configuration file, use the Log Tracking pane in the Settings dialog box, described on page 617.

The following steps describe how to manually specify your log tracking and collection preferences.

- "Prerequisites" below
- "Open the relevant log configuration file and specify your preferences" below
- "Configure the settings in the Log Tracking pane so that UFT will use the same settings that you defined in the previous step" on the next page
- "Results" on page 589

1. ### Prerequisites

   a. Your Windows-based application must use a Java or .NET log framework that includes a UDP appender.

   b. Verify the following:

      - Which log framework version is used with your application? Make sure that it includes a UDP appender and that the specific version is supported when working with UFT.

      - Make sure that logging is enabled on your application. Find out how to enable and disable logging. (You may want to enable logging for some run sessions and disable logging for others.) Do you need to define an XML parameter or modify a registry key?

      - Where is the log configuration file located? Make sure that it is writable.

2. ### Open the relevant log configuration file and specify your preferences

   This step describes how to manually configure a log configuration file so that UFT will be able to receive log messages during a run session.

   For a list of supported log frameworks, see the *HP Unified Functional Testing Product Availability Matrix*, available from the UFT Help or the root folder of the Unified Functional Testing DVD.

a. Add an **appender-ref ref** attribute with the value of QtpUdpAppender to the **root** element.

b. Specify the minimum log message level that you want to include in the run results.

   Example:

```
...
<root>
   <level value="DEBUG" />
...
   <appender-ref ref="QtpUdpAppender" />
</root>
```

c. Add an **appender** element and its attributes, as shown in the following example.

   Example:

```
<appender name="QtpUdpAppender"
 type="log4net.Appender.UdpAppender">
   <remoteAddress value="1.1.1.1" />
   <remotePort value="18081" />
   <encoding value="utf-16" />
   <layout type="log4net.Layout.XmlLayoutSchemaLog4j">
      <prefix value="" />
   </layout>
</appender>
```

> **Note:** To enable UFT to receive log messages, the **Add log messages to run results** area of the "Log Tracking Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 617 of the Settings dialog box must also be configured, as described below.

3. Configure the settings in the Log Tracking pane so that UFT will use the same settings that you defined in the previous step

This step enables UFT to receive log messages during a run session.

In the Log Tracking pane of the Settings dialog box:

- Select the "Add log messages to run results" on page 620 check box.
- Specify the settings in the upper half of the pane:
    - **Log messages source**
    - **Port**
    - **Minimum level to add node to results tree**

  For details on these fields, see "Log Tracking Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 617.
- Clear the **Auto-configure log mechanism** check box. This prevents UFT from modifying the configuration file.

4. **Results**

   During a run session, UFT receives any log messages that match or exceed the minimum log message level that you specified in the Log Tracking pane and displays them in the run results.

   In the run results tree, a node is also inserted for the first message that matches or exceeds the **Minimum level to add node to results tree**. This node is inserted directly after the step that triggered (or preceded) the relevant log message (according to its timestamp).

Reference

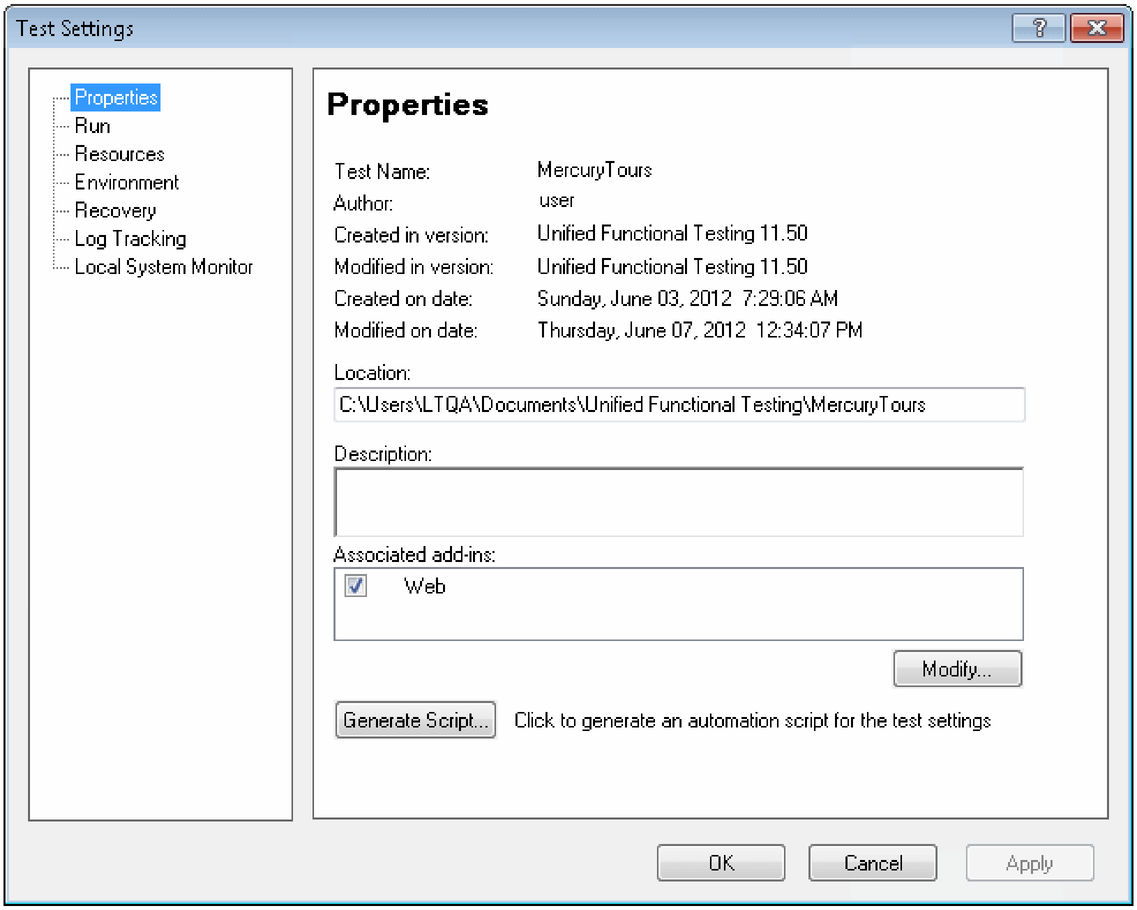
Properties Pane (Test/Business Component Settings Dialog Box)

Relevant for: GUI tests and business components

This pane enables you to view general information about your test or component, including its description and any add-ins associated with it.

For tests: You can also define the information in this pane, as well as choose to generate an automation script for the test settings.

For components: You can also set or modify the component status. All other information is displayed as defined for the application area, in read-only mode.



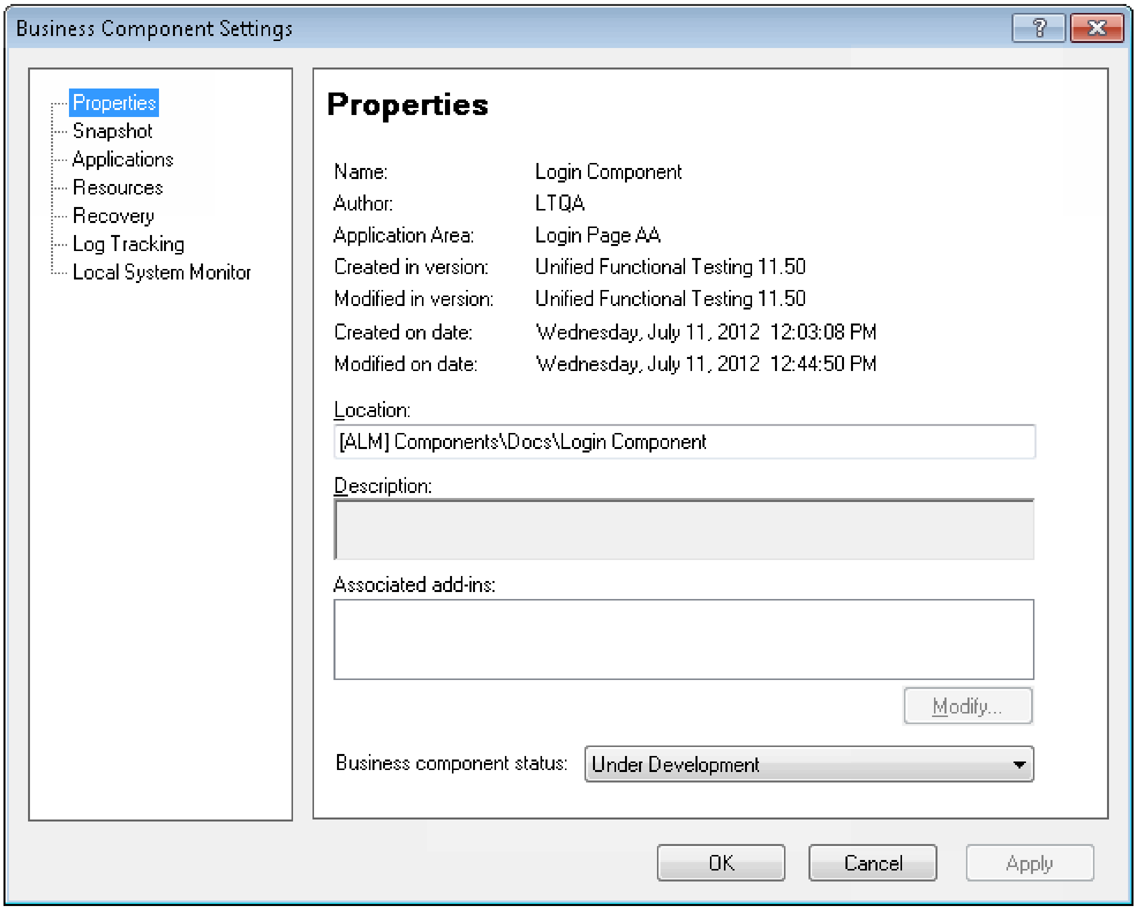
Test Name: MercuryTours
Author: user
Created in version: Unified Functional Testing 11.50
Modified in version: Unified Functional Testing 11.50
Created on date: Sunday, June 03, 2012 7:29:06 AM
Modified on date: Thursday, June 07, 2012 12:34:07 PM

Location:
C:\Users\LTQA\Documents\Unified Functional Testing\MercuryTours

Description:

Associated add-ins:
☑ Web

Click to generate an automation script for the test settings



Properties

Name: Login Component
Author: LTQA
Application Area: Login Page AA
Created in version: Unified Functional Testing 11.50
Modified in version: Unified Functional Testing 11.50
Created on date: Wednesday, July 11, 2012 12:03:08 PM
Modified on date: Wednesday, July 11, 2012 12:44:50 PM

Location:
[ALM] Components\Docs\Login Component

Description:

Associated add-ins:

Business component status: Under Development

To access	Do one of the following: - With a test, action, or business component in focus, select **File > Settings**. - For tests only: In the canvas, right-click the **Start** or **End** nodes of the test and select **Settings**.
Important Information	The information displayed in this pane is also available in the "General Properties Tab (Properties Pane - GUI Testing)" (described on page 390).
See also	For tests only: "Modify Associated Add-ins Dialog Box" on page 594

User interface elements are described below:

UI Element	Description
Name	Indicates the name of the test or component. If the test or component is saved in a version-controlled project in ALM, the version number is also shown.
Author	Indicates the Windows user name of the person who created the test.

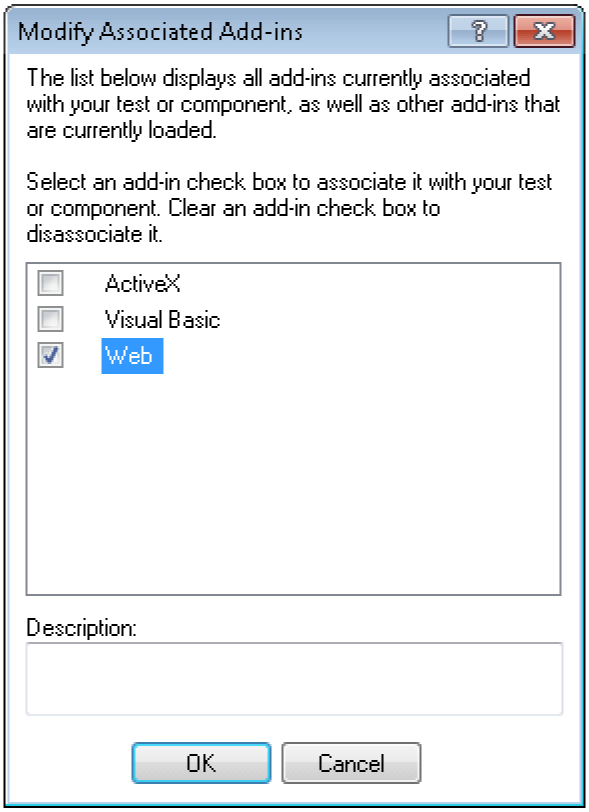
UI Element	Description
Application Area (components only)	Indicates the name of the application area which is associated with the component. For details, see "How to Create and Manage GUI Business Components" on page 2088. **Note:** If the component was created in ALM and no application area was selected, this is indicated by **Not selected**. Before business component steps can be implemented, an application area must be selected.
Created in version	Indicates the version of QuickTest or UFT used to create the test or component.
Modified in version	Indicates the version of QuickTest or UFT last used to modify the test or component.
Created on date	Indicates the date and time that the test or component was created.
Modified on date	Indicates the date and time that the test or component was last modified.
Location	Indicates the path and filename of the test, or the ALM path and filename of the component.
Description	• **For tests:** Enables you to specify a description for your test. • **For components:** Displays the description specified for your component. This field can be entered or modified only in ALM.
Associated add-ins	• **For tests:** Lists the add-ins associated with the test. For details, see "Add-in Associations in a GUI Test" on page 583. • **For components:** Lists the add-ins associated with the component (via its associated application area). The associated add-ins are loaded by business components when they are accessed.
Modify (tests only)	Enables you to select the add-ins to associate with your test. For details, see "Modify Associated Add-ins Dialog Box" on page 594.

UI Element	Description
Generate Script	Generates a UFT automation script containing the current test settings. When you click the **Generate Script** button, a Save As dialog box opens, enabling you to specify the name and file system location to store the generated file. You can use some or all of the script lines from this generated script in an automation script. This can be useful, for example, if you want to open a test with required test settings already set or to set the same test settings for multiple tests by looping through the tests in a folder. For details, see "UFT Automation Scripts" on page 1155 and the *Unified Functional Testing Automation Object Model Reference* (**Help > HP UFT GUI Testing Advanced References > Automation Object Model Reference**).
Business component Status (components only)	Specifies the status of the component. You can change the status of the component by selecting a different option from the list. For details on status options, see "Understanding GUI Component Statuses" on page 584.

Modify Associated Add-ins Dialog Box

Relevant for: GUI tests and components

This dialog box lists all the add-ins currently associated with your test, as well as any other add-ins that are currently loaded in UFT, and enables you to associate or disassociate add-ins with your test.



Modify Associated Add-ins

The list below displays all add-ins currently associated with your test or component, as well as other add-ins that are currently loaded.

Select an add-in check box to associate it with your test or component. Clear an add-in check box to disassociate it.

□ ActiveX
□ Visual Basic
☑ Web

Description:

To access	In the "Properties Pane (Test/Business Component Settings Dialog Box)", click **Modify**.

Important information	• If this dialog box contains a child add-in, and you select it, the parent add-in is selected automatically. • If you clear the check box for a parent add-in, the check boxes for its children are also cleared. • If a specific add-in is not currently loaded, but you want to associate it with your test or application area, reopen UFT and load the add-in from the Add-in Manager. If the Add-in Manager dialog box is not displayed when you open UFT, you can choose to display it the next time you open UFT. To do so, select **Display Add-in Manager on startup** from the Startup Options pane of the Options dialog box (**Tools > Options > General** tab **> Startup Options** node). For details, see "UFT Global Options" on page 522.
See also	• "Add-in Associations in a GUI Test" on page 583 • For details on the Add-in Manager, see the section on working with UFT add-ins in the *HP Unified Functional Testing Add-ins Guide*.

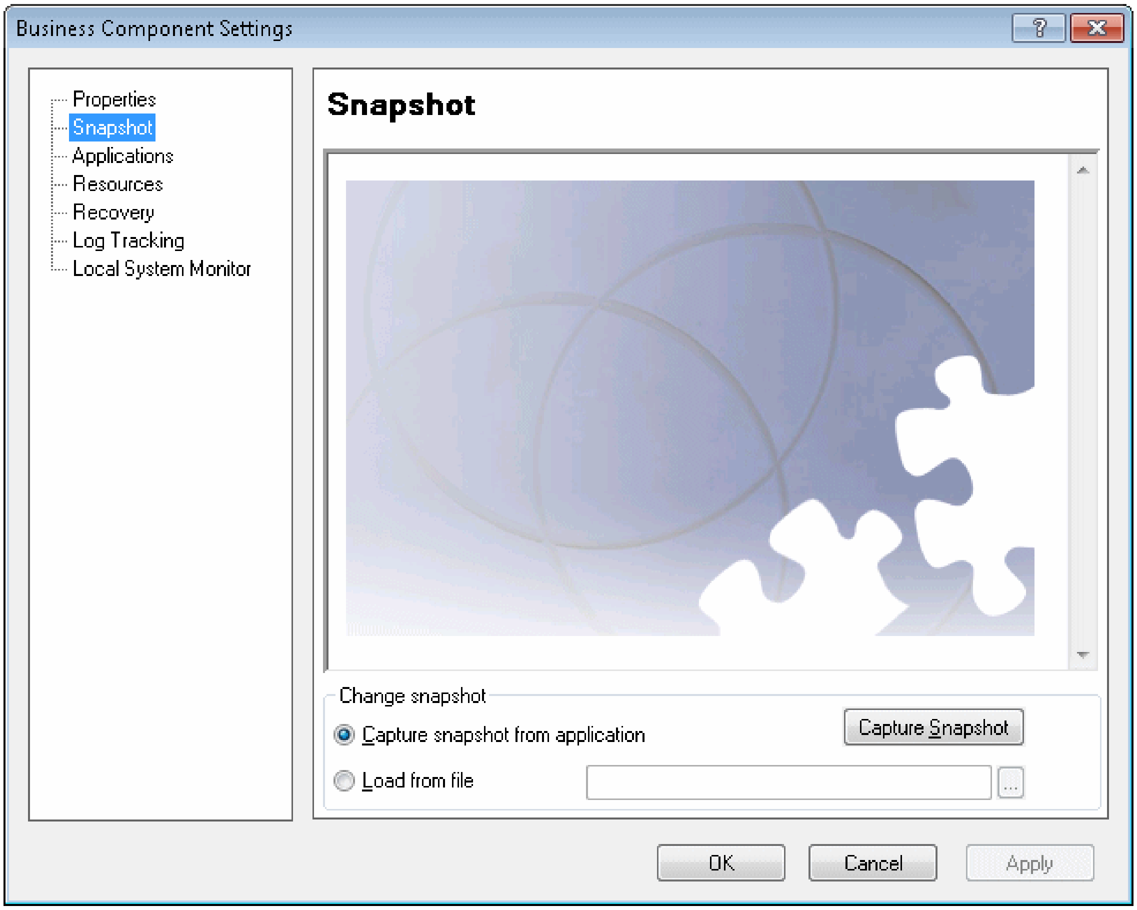
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Add-ins list>	The list of the add-ins that are currently loaded in UFT. You can select the check boxes for add-ins that you want to associate with your test or application area, or clear the check boxes for add-ins that you do not want to associate with your test or application area. In the image above: • **Web** is loaded and associated with the test or application area. • **ActiveX** is loaded, but not associated with the test or application area. • **Visual Basic** is associated with the test or application area, but is not loaded. **Note:** • Add-ins that are associated with your test or application area but not currently loaded are shown dimmed. • This list might also include child nodes representing add-ins that you or a third party developed to support additional environments or controls using add-in extensibility.
Description	The description of the selected add-in.

Snapshot Pane (Business Component Settings Dialog Box)

Relevant for: GUI components only

This pane enables you to capture or load an image and save it with the component. The image provides a visual indication of the component's main purpose. You can view the image in ALM, in the component and in any business process test in which the component is included.



To access	With a business component in focus, select **File > Settings > Snapshot**.
Important information	The snapshot image can also be captured and saved with the component from the Snapshot tab in ALM when installed with BPT support. For details on capturing a snapshot for a component in ALM, see the *HP Business Process Testing User Guide*.

User interface elements are described below:

UI Elements	Description
Capture snapshot from application	Enables you to define the image to be captured by clicking the **Capture Snapshot** button. You can then drag the crosshairs pointer to select the area to be captured. When you release the mouse button, the captured area is displayed in the Snapshot pane.
Load from file	Specifies the .png or .bmp file containing the required image. You can enter the path and filename or use the browse button to locate the file.

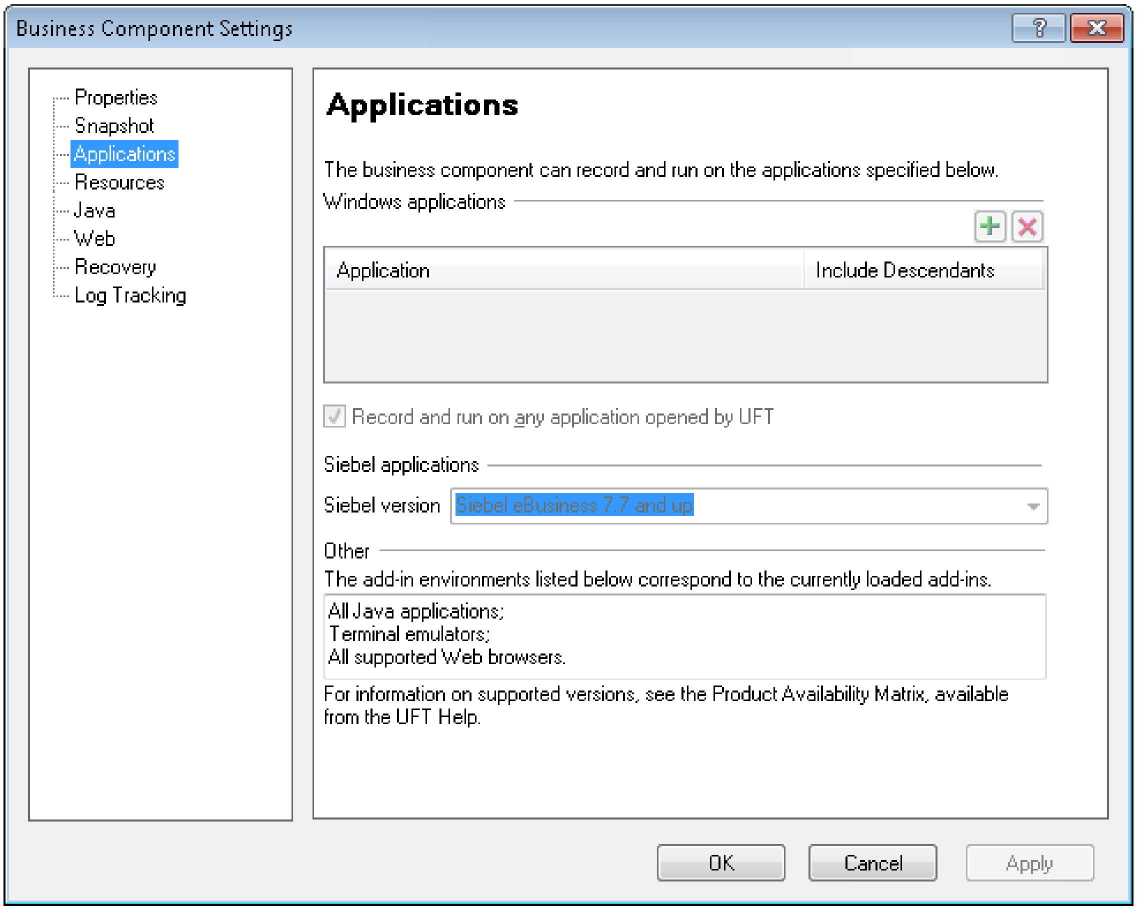
Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)

Relevant for: GUI components only

Application Areas: This pane enables you to specify the Windows-based applications on which the components associated with the application area can record and run. It also displays the environments on which the component can currently record (based on the currently loaded add-ins).

Business Components: When viewing this pane from the Business Component Settings dialog box, the options are read-only.

The following image is an example of the **Applications** pane in the Business Component Settings dialog box. These settings are identical to the ones available in an application area's Additional Settings pane.



Applications

The business component can record and run on the applications specified below.

Windows applications

Application	Include Descendants

☑ Record and run on any application opened by UFT

Siebel applications

Siebel version: Siebel eBusiness 7.7 and up

Other

The add-in environments listed below correspond to the currently loaded add-ins.

All Java applications;
Terminal emulators;
All supported Web browsers.

For information on supported versions, see the Product Availability Matrix, available from the UFT Help.

To access	Do one of the following: - For a business component: With a component in focus, select **File > Settings > Applications** node. - For an application area: Open the application area and select **Additional Settings > Applications** in the sidebar.

Important information	• You can use the Applications pane to set or modify your application preferences in the following scenarios: ▪ You have already recorded one or more steps in an associated component and you want to modify the settings before you continue recording. ▪ You want to record and run the component on a different application than the one you previously used. • You can record component steps only on the specified applications. • If you are recording a new component, set your application settings in the Applications pane of the Application Area Additional Settings using the Applications dialog box. This dialog box opens when you start to record, and contains the same options as the Applications pane, described in this section. • The Applications dialog box and Applications pane may also contain options applicable to any UFT add-ins installed on your computer. For details regarding these options, see the documentation provided for the specific add-in.
See also	"Select Application Dialog Box" on the next page

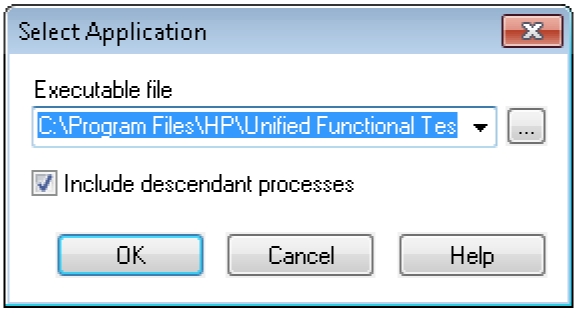
User interface elements are described below:

UI Element	Description
Windows applications	Lists the details of the Windows-based applications on which to record and run components associated with this application area. For details on the details displayed, see "Select Application Dialog Box" on the next page. If you do not want to record or run on Windows-based applications, leave the application list blank. (This is the default setting.)
✚	Adds an application to the application list. You can add up to ten applications. For details, see "Select Application Dialog Box" on the next page.
✖	Removes the selected application from the **Application** list.
Record and run on any applications opened by UFT	Records and runs on any applications invoked by UFT (as child processes of UFT), for example, applications opened during a record or run session using an OpenApp function.
Other	Displays the environments on which the application area's associated components can currently record (based on the currently loaded add-ins). For details, see the *HP Unified Functional Testing Add-ins Guide*.

Select Application Dialog Box

Relevant for: GUI components only

This dialog box enables you to select applications for use in your application area.



Executable file: C:\Program Files\HP\Unified Functional Tes

☑ Include descendant processes

To access	In the "Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" (described on page 597), click the **Add** ➕ button.
Important information	• You can add up to ten applications to the application list displayed in the Applications pane, and you can edit an existing application in the list. You can also select whether to record and run on the application's descendant processes. • The details entered in the Select Application dialog box are displayed as a single line for each application in the **Windows-based applications** area of the Applications pane.

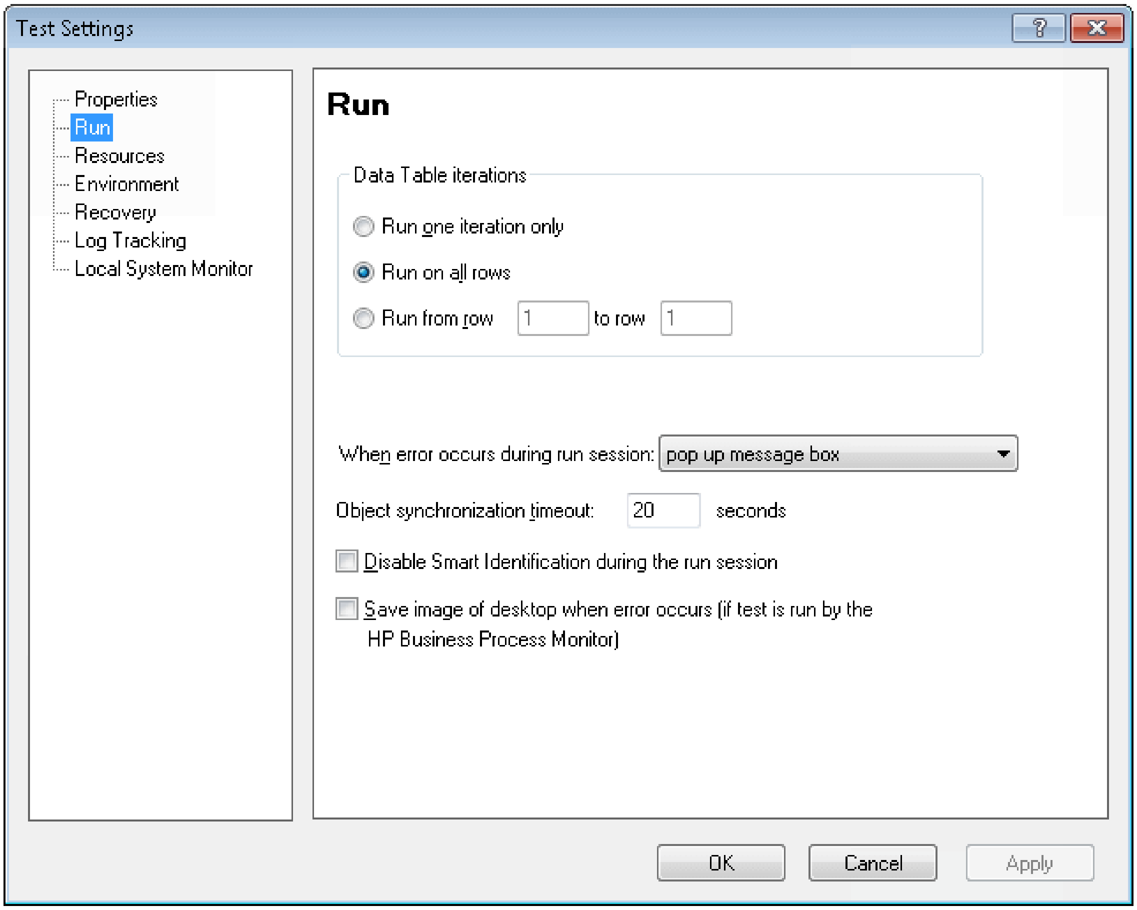
User interface elements are described below:

UI Element	Description
Executable file	Instructs UFT to record and run on the specified executable file.
Include descendant processes	Instructs UFT 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 components on this application, otherwise the functionality will not be recorded, or the run session will fail. **Default value:** Selected.

Run Pane (Test Settings Dialog Box)

Relevant for: GUI tests only

This pane enables you to choose what to do when an error occurs during the run session, set the object synchronization timeout, and choose whether or not to disable the Smart Identification mechanism for the test.



Data Table iterations

○ Run one iteration only

● Run on all rows

○ Run from row 1 to row 1

When error occurs during run session: pop up message box

Object synchronization timeout: 20 seconds

□ Disable Smart Identification during the run session

□ Save image of desktop when error occurs (if test is run by the HP Business Process Monitor)

To access	With a test or an action in focus, select **File > Settings > Run**.
Important information	• This pane applies to the entire test. You can set the run properties for an individual action in a test from the Run tab in the Action Call Properties dialog box of a selected action. For details on action run properties, see "Run Tab (Action Call Properties Dialog Box)" on page 895. • By default, when you run a test with global data table parameters, UFT runs the test for each row in the Data pane using the parameters you specified. For details, see "When to Choose Global or Action Data Table Parameters " on page 1538. • You can use this pane to instruct UFT to run iterations on a test only for certain lines in the Global tab in the Data pane.

User interface elements are described below:

UI Element	Description
Data Table iterations	Specifies the iterations for the test. Select an option: • **Run one iteration only.** Runs the test only once, using only the first row in the global data table. • **Run on all rows.** Runs the test with iterations using all rows in the global data table. • **Run from row __ to row __.** Runs the test with iterations using the values in the global data table for the specified row range.
When error occurs during run session	Specifies how UFT responds to an error during the run session. For details on the different responses, see "Error Response Options" on the next page.
Object synchronization timeout	Sets the maximum time (in seconds) that UFT waits for an object to load before running a step in the test. **Note:** When working with Web objects, UFT waits up to the amount of time set for the **Browser navigation timeout** option, plus the time set for the object synchronization timeout. For details on the **Browser navigation timeout** option, see the *HP Unified Functional Testing Add-ins Guide*.
Disable Smart Identification during the run session	Instructs UFT not to use the Smart Identification mechanism during the run session. **Note:** When you select this option, the **Enable Smart Identification** check boxes in the Object Properties and Object Repository windows are disabled, although the settings are saved. When you clear this option, the **Enable Smart Identification** check boxes return to their previous on or off setting.
Save image of desktop when error occurs (if test is run by the HP Business Process Monitor)	This option is applicable only to tests that are run by the Business Process Monitor component of HP Business Availability Center. Selecting this option instructs UFT to capture a snapshot of the desktop if an error occurs during a run session of a test initiated by the Business Process Monitor. The image is saved in Business Availability Center. The Business Process Monitor forwards the run results to the Business Availability Center servers.

Error Response Options

Relevant for: GUI tests only

By default, if an error occurs during the run session, UFT displays a popup message box describing the error. You must click a button on this message box to continue or end the run session.

In the Run pane of the Test Settings dialog box, you can accept the **popup message box** option or you can specify a different response by choosing one of the alternative options in the list in the **When error occurs during run session** box:

- **proceed to next action iteration.** UFT proceeds to the next action iteration when an error occurs.
- **stop run.** UFT stops the run session when an error occurs.
- **proceed to next step.** UFT proceeds to the next step in the test when an error occurs.

> **Note:** If you accept the **popup message box** option, UFT also continues to the next step of the run.

UFT first performs any recovery scenarios associated with the test, and performs the option selected above only if the associated recovery scenarios do not resolve the error. For details, see "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 613.

> **Note:** If you are working with many tests, you may want to use a UFT automation script to set a different value for each test. To access the automation script line that controls this option, you can use the **Generate Script** button in the Properties pane of the Test Settings dialog box.
>
> For details, see "UFT Automation Scripts" on page 1155 or the *HP UFT Object Model Reference for GUI Testing* (**Help > HP UFT GUI Testing Advanced References Help > HP Unified Functional Testing Automation Object Model Reference**).

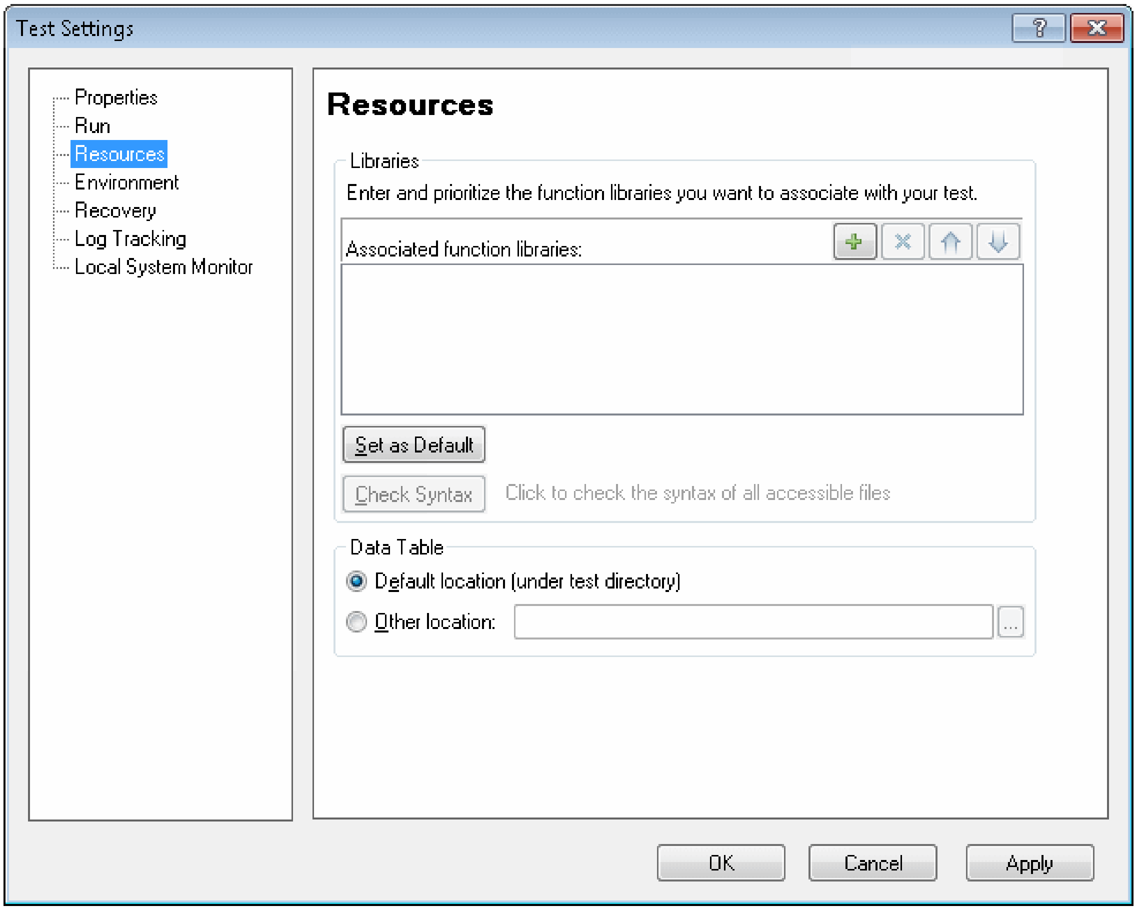
Resources Pane (Test/Business Component Settings Dialog Box)

Relevant for: GUI tests and components

This pane displays the list of resources associated with your test or component (component associations are made via the component's application area).

For tests: The pane enables you to associate function libraries and data table files with your test. You can also set the currently associated function library settings as the default settings for all new tests.

For components: The pane displays a list of the function libraries and object repositories associated with the component via its application area, in read-only mode.

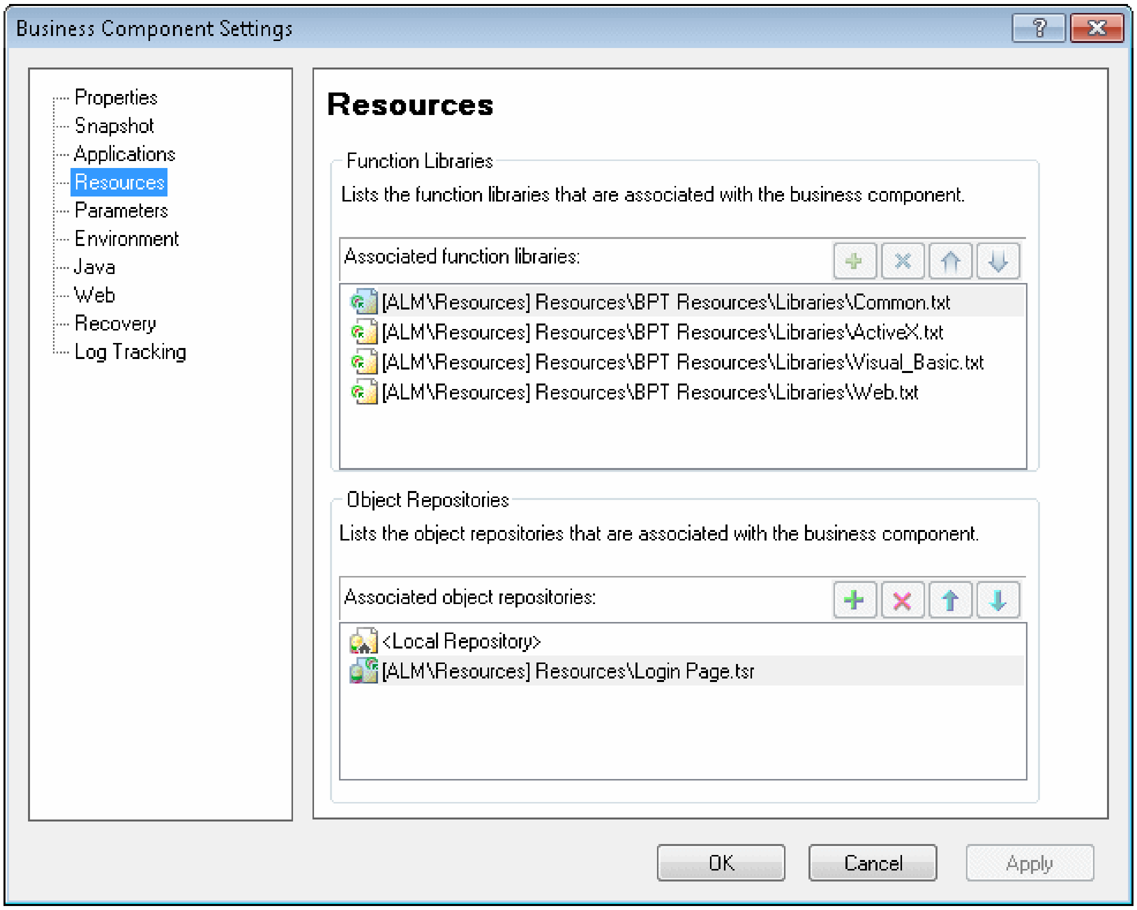


Enter and prioritize the function libraries you want to associate with your test.

Associated function libraries:
Set as Default
Check Syntax
Click to check the syntax of all accessible files
Data Table

● Default location (under test directory)

○ Other location:



Lists the function libraries that are associated with the business component.

Associated function libraries:

- [ALM\Resources] Resources\BPT Resources\Libraries\Common.txt
- [ALM\Resources] Resources\BPT Resources\Libraries\ActiveX.txt
- [ALM\Resources] Resources\BPT Resources\Libraries\Visual_Basic.txt
- [ALM\Resources] Resources\BPT Resources\Libraries\Web.txt

Lists the object repositories that are associated with the business component.

Associated object repositories:

- <Local Repository>
- [ALM\Resources] Resources\Login Page.tsr

To access	With a test, action, or business component in focus, select **File > Settings > Resources**.
Important information	**For tests:** • Object repositories are associated with individual actions in your test. You can associate an object repository with an action using the Action Properties dialog box (right-click an action in the canvas and select **Action Properties**) and the Associate Repositories dialog box (**Resources > Associate Repositories**). • When running a test, UFT uses associated function libraries from ALM project folders only when you are connected to the corresponding ALM project. For details, see "ALM Integration" on page 707. **For components:** The information in this pane is read-only. You define the settings for this pane using the "Function Libraries Pane (Application Area)" on page 2102 and the "Object Repositories Pane (Application Area)" on page 2105.
See also	• "Associated Function Libraries" on page 1031 • "Function Libraries Pane (Application Area)" on page 2102 (for components)

User interface elements are described below:

UI Element	Description
Associated function libraries	The list of function libraries associated with your test or component. The order of the function libraries in the list determines the order in which UFT searches for a function or subroutine that is called from a step in your test or component. **For tests:** You can add, delete, and prioritize the files. You can also set the default function libraries for new tests.
➕	Associates a function library with the test. (If the function library contains syntax errors, a message opens stating that your test or component will fail because of these syntax errors.) You can enter the absolute or relative path and filename of the function library, or use the browse button to locate the required file. The function library can be located in the file system or in an ALM project folder. If you enter a relative path, then during a 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. For details, see "Relative Paths in UFT for GUI Testing" on page 130. **Note:** • If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, specify an absolute Quality Center or ALM path. For details, see "Relative Paths and ALM" on page 758. • When you are connected to ALM and, UFT adds [ALM], and displays a browse button so that you can locate the ALM path. • When not connected to ALM, you can add a file located in an ALM project folder by holding the SHIFT key and clicking this button. UFT adds [ALM], and you can enter the path. You can also type the entire ALM path manually. If you do, you must add a space after [ALM]. For example: [ALM] Subject\Tests.
❌	Removes an associated function library from the list.
⇧	Moves the selected function library up in the list, assigning it a higher priority.
⇩	Moves the selected function library down in the list, assigning it a lower priority.

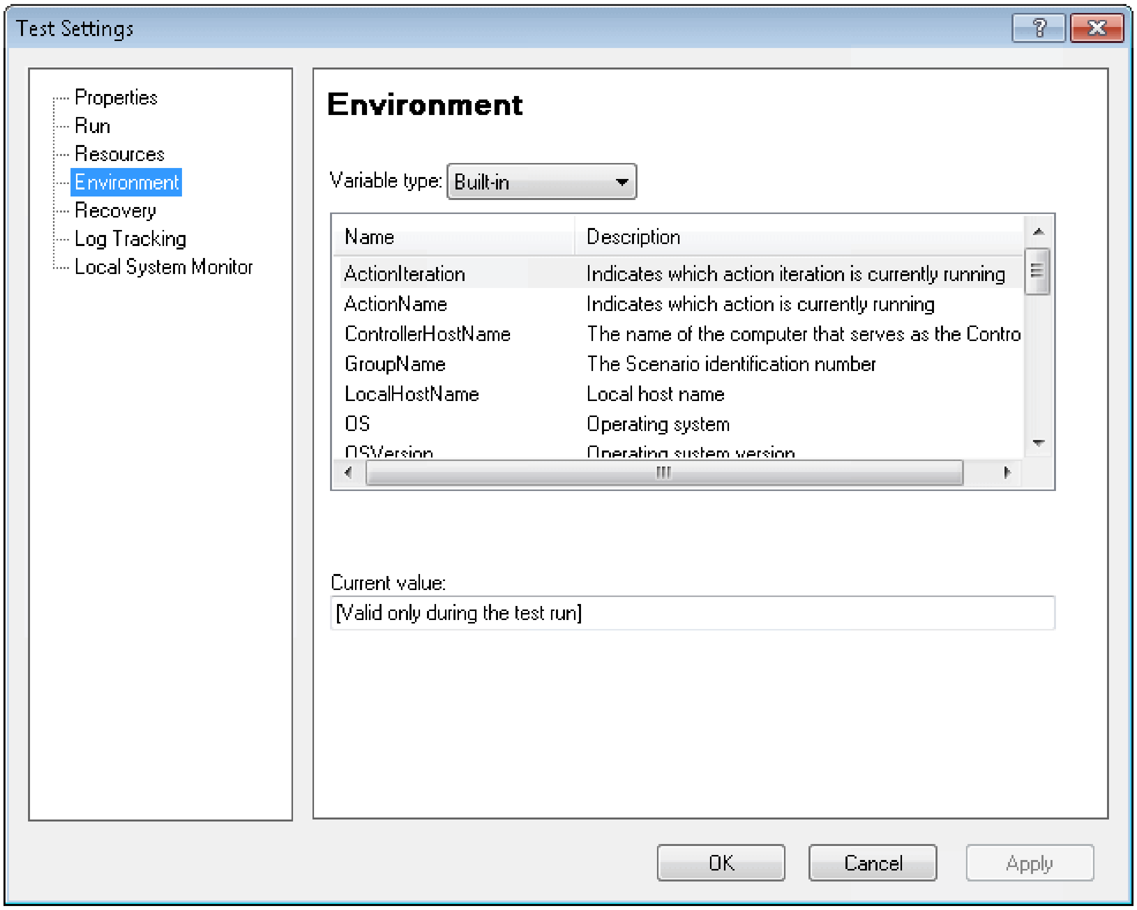
UI Element	Description
Set as Default (tests only)	Sets the current list of function libraries as the default list to be associated with new tests. This does not affect existing tests. **Note:** This button is enabled when the list of associated function libraries for this test is different from the list that was previously set as default for all tests. **Caution:** If the default function library is moved or renamed, UFT will not be able to locate it. The function library will be displayed in the Errors pane when new tests are created. For details on resolving missing resources, see "Errors Pane" on page 364.
Check Syntax (tests only)	Verifies whether any of the associated function libraries contain syntax errors that will prevent the test from running properly. Click the **Check Syntax** button to check the files for syntax errors before finalizing the test. If any syntax errors are found, the Errors pane opens listing the files containing syntax errors. Otherwise, an information box opens confirming that the syntax in all of the function libraries is valid. **Note:** • UFT checks only the associated function libraries that can be accessed. For example, if an associated function library is stored in an ALM project to which you are not currently connected, its syntax will not be checked. • You cannot instruct UFT to check syntax for function libraries that are loaded dynamically during a run.
Data Table (tests only)	The location of the data table to be used in your test: • **Default location (under test folder).** Instructs UFT to use data stored in the default data table location under the test folder. • **Other location.** Instructs UFT to use data stored in the specified data table location. The data table can be any Microsoft Excel (.xls or .xlsx) file. For details on data tables, see "Data Pane Overview" on page 222. **Note:** You can specify Microsoft Excel files stored in ALM as data tables. For details, "How to Manage Data Tables in a GUI Test" on page 231.

UI Element	Description
Associated object Repositories (components only)	The list of shared object repositories currently associated with your component via its associated application area (in addition to the local object repository). The order of the object repositories in the list determines the order in which UFT searches for a test object description. Components use shared object repository files stored in ALM. For details on associating object repositories with application areas, see "Object Repositories Pane (Application Area)" on page 2105.

Environment Pane (Test Settings Dialog Box)

Relevant for: GUI tests and scripted GUI components

This pane displays existing built-in and user-defined environment variables. It also enables you to add, modify, or delete internal user-defined environment variables, export the defined variables to an external .xml file, and retrieve them from a file.

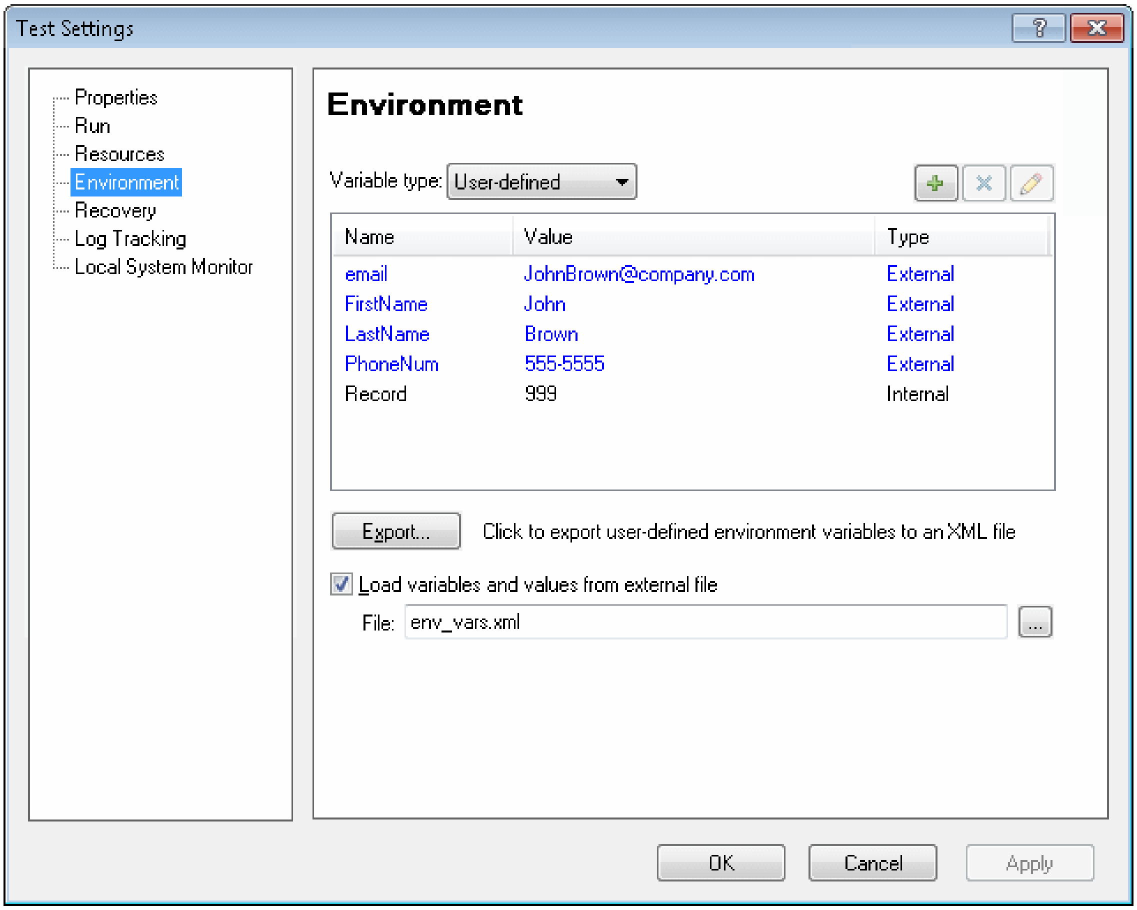


Environment

Variable type: Built-in

Name	Description
ActionIteration	Indicates which action iteration is currently running
ActionName	Indicates which action is currently running
ControllerHostName	The name of the computer that serves as the Contro
GroupName	The Scenario identification number
LocalHostName	Local host name
OS	Operating system
OSVersion	Operating system version

Current value:
[Valid only during the test run]



Variable type: User-defined

Name	Value	Type
email	JohnBrown@company.com	External
FirstName	John	External
LastName	Brown	External
PhoneNum	555-5555	External
Record	999	Internal

Click to export user-defined environment variables to an XML file

☑ Load variables and values from external file

File: env_vars.xml

To access	With a test, action, or scripted component in focus, select **File > Settings > Environment**.
Important information	Variables from an external environment variables file are displayed in blue. Internal environment variables are displayed in black.
See also	"Environment Variable Parameters" on page 1536

This pane includes the following key areas:

- "Variable Type Selection Area" on the next page
- "Built-in Environment Variables Area" on the next page
- "User-Defined Environment Variables Area" on the next page

Variable Type Selection Area

UI Element	Description
Variable type	Enables you to select either a **Built-in** environment variable or a **User-defined** environment variable to define.

Built-in Environment Variables Area

Relevant for: GUI tests and scripted GUI components

Displays the built-in environment variables defined by UFT and their current values.

This area provides the following information:

UI Element	Description
Name	The name of each built-in environment variable.
Description	A short description of each built-in environment variable.
Current value	The current value of the selected environment variable.

User-Defined Environment Variables Area

Relevant for: GUI tests only

Displays both internal and external user-defined environment variables and their current values.

This area provides the following information:

UI Element	Description
Name	The name of each user-defined variable.
Value	The value assigned to each user-defined variable.
Type	The type of each user-defined variable: Internal or External. Internal environment variables are available only to the test in which they are defined.
➕	Enables you to define a new internal environment variable and add it to the list. For details, see "Add New/View/Edit Environment Parameter Dialog Box" on the next page.
✖	Deletes a selected internal environment variable from the list. **Note:** After you confirm the deletion of the environment variable, you cannot retrieve it, even if you click **Cancel** in the Test Settings dialog box.

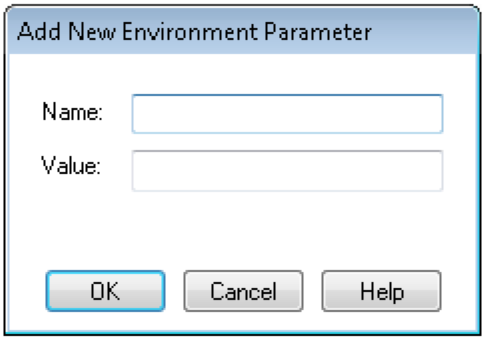
UI Element	Description
✎	Enables you to edit the value of a selected internal environment variable or to view the properties of a selected external environment variable. For details, see "Add New/View/Edit Environment Parameter Dialog Box" below.
Export	Exports your user-defined environment variables to an external .xml file for use with other tests. You can then use the exported environment variable file with any test by loading them from the file as external user-defined environment variables. For details, see "Save <Resource>/Save <Document> As Dialog Box" on page 164. **Note:** • If the file is saved to the file system, its values are loaded each time the test runs. If the file is saved to an ALM project, its values are loaded when the test is first loaded. • If the values are changed after the test is loaded, UFT will not use the new values until the next time the test is loaded. • If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, you should specify an absolute Quality Center or ALM path. For details, see "Relative Paths and ALM" on page 758.
Load variables and values from external file	Instructs UFT to load the variables saved in the .xml file that you specify for use with your test.
File	The environment variable .xml file to load and use when your test runs. Click the **Browse** button to open the Open XML File dialog box. For details, see "Open/New <Document>/<Resource> Dialog Box" on page 156.

Add New/View/Edit Environment Parameter Dialog Box

Relevant for: GUI tests only

This dialog box enables you to add, edit and internal user-defined environment variables from the Environment pane of the Test Settings dialog box. For external user-defined environment variables, you can view the variable details in read-only format.

The following image is an example of the dialog box when adding a new environment parameter. The actual name of the dialog box may differ depending on the operation you are performing.



To access	In the user-defined environment variable area of the "Environment Pane (Test Settings Dialog Box)""Environment Pane (Test Settings Dialog Box)", do one of the following: - **To add:** Click the **New** ✚ button. - **To view, modify, or copy:** Select the environment variable and click the **View/Edit Environment Variable** button ✎ .
Important information	Internal environment variables are available only to the test in which they are defined.

User interface elements are described below:

UI Elements	Description
Name	The name of the variable.
Value	The value of the variable. **Tip:** To copy the value of the variable to the Clipboard, select the value text, right-click, and select **Copy**.
📝	**View/Edit Complex Value**. Enables you to view the contents of the value in a multi-line edit box. Available only when the value cannot be displayed entirely in the **Value** box.

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

Relevant for: GUI tests and components

This pane displays a list of all recovery scenarios associated with the current test or application area. Recovery scenario settings enable you to specify how a test or business component recovers from unexpected events and errors during a run session.

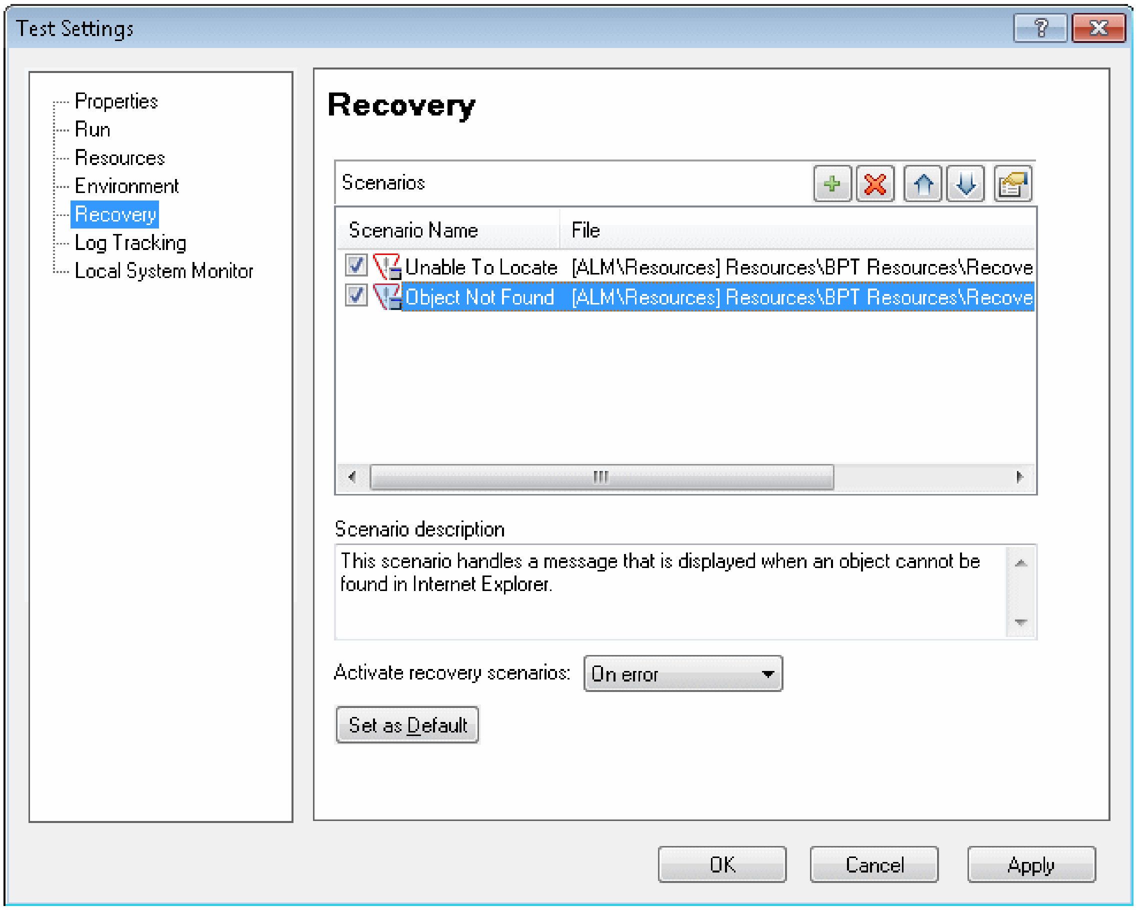
This pane also enables you to associate additional recovery scenarios with the test or application area, create recovery scenarios, remove scenarios from the test or application area, change the order in which they are applied to the run session, and view a read-only summary of each scenario.

For tests: You can also specify the default list of scenarios to associate with all new tests.

For components: This pane displays the recovery information defined for the application area, in read-only mode.

The following image shows the **Recovery** pane in the Test Settings dialog box.

The options and the panes available in the sidebar differ slightly in an application area's Additional Settings pane and in the Business Component Settings dialog box.



	Scenario Name	File
☑	Unable To Locate	[ALM\Resources] Resources\BPT Resources\Recove
☑	Object Not Found	[ALM\Resources] Resources\BPT Resources\Recove

Scenario description

This scenario handles a message that is displayed when an object cannot be found in Internet Explorer.

Activate recovery scenarios: On error

To access	Do one of the following: • For a test or business component: With a test, action, or component in focus, select **File > Settings > Recovery** node. • For an application area: Open the application area and select **Additional Settings > Recovery** in the sidebar.
Important information	**For tests:** • The default recovery scenarios provided with UFT are installed in your UFT installation folder. The paths specifying the default recovery scenarios in the Recovery pane use an environment variable (%ProductDir%) in the file path. This enables UFT to locate these recovery scenarios when tests associated with them are run on different computers or by different HP products. Do not modify the file paths of these default recovery scenarios or attempt to use the environment variable for any other purpose. • You can also associate, remove, enable, disable, prioritize, and view the properties of the recovery scenarios associated with your test in the Solution Explorer. For details, see "Solution Explorer Pane" on page 470. **For components:** • The options in this pane are read-only. You define the recovery scenario settings for a component in its associated application area. • UFT provides you with a sample recovery file for Web-related testing. The file is located in the Test Resources module of your ALM project under Resources\BPT Resources\Recovery Scenarios\DefaultWeb.qrs.
See also	"Recovery Scenarios for GUI Testing" on page 1111

This pane includes the following key elements:

- "Scenarios Area" on the next page
- "Scenario Description and General Options Area" on page 616
- "Scenario Type Icons" on page 616

Scenarios Area

UI Element	Description
➕	Opens the Add Recovery Scenario dialog box, which enables you to associate one or more recovery scenarios with the test or application area. For details, see "Add Recovery Scenario Dialog Box" on page 1120.
✖	Removes the selected recovery scenario from the test or application area.
⇧	Moves the selected scenario up in the list, giving it a higher priority.
⇩	Moves the selected scenario down in the list, giving it a lower priority.
🗒	Displays summary properties for the selected recovery scenario in read-only format. For details, see "Recovery Scenario Properties Dialog Box" on page 1124.
Scenario Name	The name of each recovery scenario associated with your test or application area. You can add, delete, and prioritize the scenarios in the list.
File	The file path for each recovery scenario associated with your test or application area. You can edit the recovery scenario file path by clicking the path once to highlight it, and then clicking it again to enter edit mode. For details on scenario type icons, see "Scenario Type Icons" on the next page. For example, you may want to modify an absolute file path to be a relative file path. If you modify a recovery scenario file path, ensure that the recovery scenario exists in the new path location before running your test or component. **For tests:** • If your recovery files are stored in the file system and you want other users or HP products to be able to run this test on other computers, you should set the recovery file path as a relative path. (Click the path once to highlight it, and then click it again to enter edit mode.) Any users who want to run this test should then specify the drive letter and folder in which UFT should search for the relative path in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node). For details, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544 and "Relative Paths in UFT for GUI Testing" on page 130. • If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, you should store your recovery files in the Test Resources module and specify an absolute Quality Center or ALM path in the Folders pane. For details, see "Relative Paths and ALM" on page 758.

Scenario Description and General Options Area

UI Elements	Description
Scenario description	Displays the textual description of the scenario selected in the Scenarios box. You can select or clear the check box next to each scenario to enable or disable it for the current test or application area.
Activate recovery scenarios	Indicates how often UFT should activate the recovery mechanism: • **On every step.** The recovery mechanism is activated after every step. • **On error.** The recovery mechanism is activated only after steps that return an error return value. • **Never.** The recovery mechanism is disabled. **Note:** Choosing **On every step** may result in slower performance during the run session.
Set as Default (tests only)	Sets the current list of recovery scenario files as the default list to be associated with new tests. **Note:** The **Set as Default** option is available for tests only. It is enabled when the setting for this test is different from the default for all tests. **Note:** If the file containing the recovery scenarios is moved or renamed, UFT will not be able to locate it. The recovery scenario file will be displayed in the Errors pane when new actions or tests are created. For details on resolving missing resources, see "Errors Pane" on page 364.

Scenario Type Icons

Icon	Description
	Indicates that the recovery scenario is triggered by a specific pop-up window in an open application during the run session.
	Indicates that the recovery scenario is triggered when the property values of an object in an application match specified values.
	Indicates that the recovery scenario is triggered when a step in the test or component does not run successfully.
	Indicates that the recovery scenario is triggered when a specified application fails during the run session.

Icon	Description
🗎	Indicates that the recovery scenario is no longer available for the test or component—possibly because the recovery file has been renamed or moved, or can no longer be accessed by UFT. When an associated recovery file is not available during a run session, a message is displayed in the run results.

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

Relevant for: GUI tests and components

This pane enables you to configure your log tracking and collection preferences.

This pane enables you to view your log tracking and collection preferences.

For tests: Use this pane to configure your preferences.

For components: This pane displays the log tracking information defined for the application area, in read-only mode.

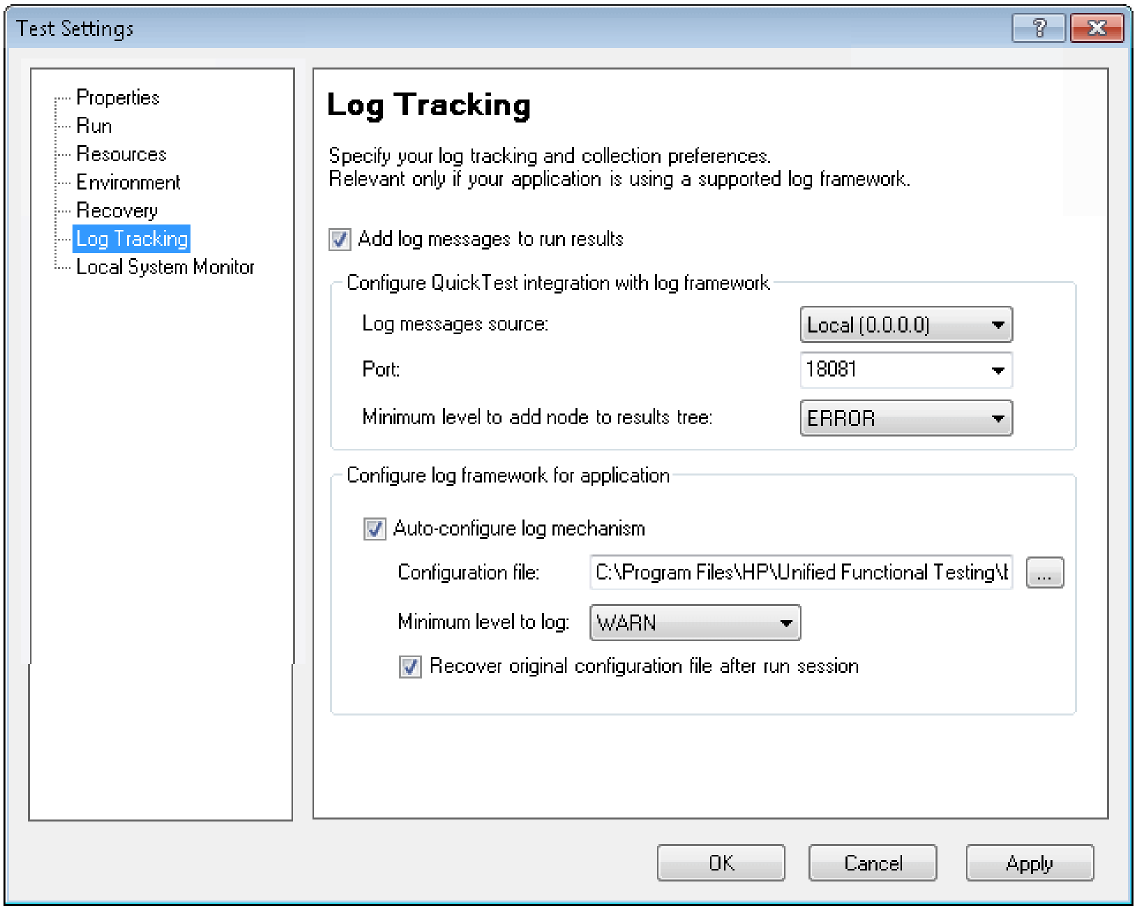
This pane is divided into two sections:

- The settings in the top half are UFT-specific, enabling it to receive log messages during a run session.
- The settings in the bottom half are specific to the log configuration file used by your application.

When configured, these settings are used during a run session if your Windows-based application uses a Java or .NET log framework that includes a UDP appender. The log messages generated by your application are displayed in the run results. You can use this information to detect unexpected behavior in your application.

The following image shows the Log Tracking pane in the Test Settings dialog box.

The panes available in the sidebar differ slightly in an application area's Additional Settings pane and in the Business Component Settings dialog box.



Log Tracking

Specify your log tracking and collection preferences.
Relevant only if your application is using a supported log framework.

☑ Add log messages to run results

Configure QuickTest integration with log framework

Log messages source: Local (0.0.0.0)

Port: 18081

Minimum level to add node to results tree: ERROR

Configure log framework for application

☑ Auto-configure log mechanism

Configuration file: C:\Program Files\HP\Unified Functional Testing\l ...

Minimum level to log: WARN

☑ Recover original configuration file after run session

To access	Do one of the following: - For a test or business component: With a test, action, or component in focus, select **File > Settings > Log Tracking** node. - For an application area: Open the application area and select **Additional Settings > Log Tracking** in the sidebar.

Important information	Prerequisites: • Make sure that the firewall does not block the UDP port. • The log framework must use an XML-based configuration file. (This enables you to configure it to send log messages to UFT.) • UFT uses timestamps to associate log messages with the relevant test or component step. Therefore, if your logging application is located on a remote computer, make sure that the system clocks on your application's computer (where UFT is installed) and the remote computer are synchronized. • Some applications may need to be restarted after modifying the log configuration file. Limitations: • These settings cannot be modified during a run session. • If external events affect your application while it is being tested, messages about these events may also be sent to the run results. • **Caution:** To ensure that log tracking settings are saved for application areas, make sure that you make at least one other modification after modifying the **Minimum level to log** settings. If necessary, select a checkbox and then clear it. Settings will not be saved if the Minimum level to log is the last setting defined. • **For business process tests:** Log tracking runs only if the application area associated with the first component in the business process test enables it. This means that if another component that runs later in the business process test is associated with a different application area, log tracking settings from that application area are ignored.
Relevant tasks	"How to Manually Configure Log Tracking Settings" on page 587
See also	• "Log Tracking" on page 586 • The section on the Log Tracking pane in the *HP Run Results Viewer User Guide*

User interface elements are described below:

UI Elements	Description
Add log messages to run results	Enables UFT to receive log messages from the log framework used by your application and to include these messages in the run results. In the Run Results viewer, these messages are displayed in the Log Tracking Results pane. You can click a log message to locate the step in the run results tree at which the event that preceded/triggered the message occurred (according to its timestamp). For details, see the section on the Log Tracking pane in the *HP Run Results Viewer User Guide*.
Log messages source	• **Network (0.0.0.0).** (Default) Enables UFT to receive log messages from the log framework wherever it is located—either on a remote computer on the network or on the local computer. • **Local (127.0.0.1).** Enables UFT to receive log messages only from the log framework on the local computer.
Port	The port that UFT listens to on the computer on which the log framework runs. You can select any UDP port that is not in use.
Minimum level to add node to results tree	The minimum log message level for which a node is added to the run results. A node is inserted in the run results tree directly following each step that triggers a log message that matches or exceeds the value you select in this option. You can view details for these message in the Result Details pane. You can also view the generated messages in the Results.xml file in the <Result Name>\Report folder. Possible values in order of severity: • **TRACE** (maps to **micDone**) • **DEBUG** (maps to **micDone**) • **INFO** (maps to **micPass**) • **WARN** (maps to **micWarning**) • **ERROR** (maps to **micFail**) • **FATAL** (maps to **micFail**) Default: **ERROR**

UI Elements	Description
Auto-configure log mechanism	Instructs UFT to configure the log framework (via its log configuration file) at the beginning of every run session according to the settings in this pane. If you clear this check box, UFT receives log messages only if you modify the log configuration file accordingly. For details, see "How to Manually Configure Log Tracking Settings" on page 587. When using the **Auto-configure log mechanism** option, make sure that: • All logging prerequisites are met, for example, you may need to set registry values that are specific to your application. • The log framework's configuration file is writable and is in a location that UFT can access. • Your application's log framework is configured to use the same file you specified in the "Configuration file" edit box. • Your application's log framework can monitor changes in the configuration file. If not, then you must always start your application after the run session begins so that UFT can modify the configuration file. For details on starting your application, see the **SystemUtil.Run Method** in the **Standard Windows** section of the *HP UFT Object Model Reference for GUI Testing*. You can also use any standard VB Script command that starts an application. **Note:** If your application's log framework monitors your configuration file infrequently (for example, once per minute), start your application immediately after the run session begins. This enables UFT to modify the configuration file in time to receive the log messages generated during the run session.

UI Elements	Description
Configuration file	The root path of the configuration file used by the log mechanism. The configuration file can be stored in any accessible location. Possible configuration file types: • ***.XML** (Java and .NET) • ***.CONFIG** (.NET) • ***.LOG4NET** (.NET) **Note:** UFTcan update only one file per run session, so if there are multiple configuration files (for example, if the test or component is validating more than one application - each with its own configuration file), UFT modifies only the file specified in this text box. If you need to configure multiple files, do so manually. UFT can receive log messages from multiple applications, but can auto-configure only one of them.
Minimum level to log	The minimum log message level that UFT receives from the log framework and sends to the run results. These log messages are displayed in the Log Tracking Pane in the Run Results Viewer and can also be viewed in the LogMessage.xml file in the <Result Name>\Report folder. Possible values in order of severity: • **TRACE** (maps to **micDone**) • **DEBUG** (maps to **micDone**) • **INFO** (maps to **micPass**) • **WARN** (maps to **micWarning**) • **ERROR** (maps to **micFail**) • **FATAL** (maps to **micFail**) Default: **WARN**
Recover original configuration file after run session	Restores the configuration file that existed prior to the beginning of the run session (instead of keeping the configuration file modified at the beginning of the run session).

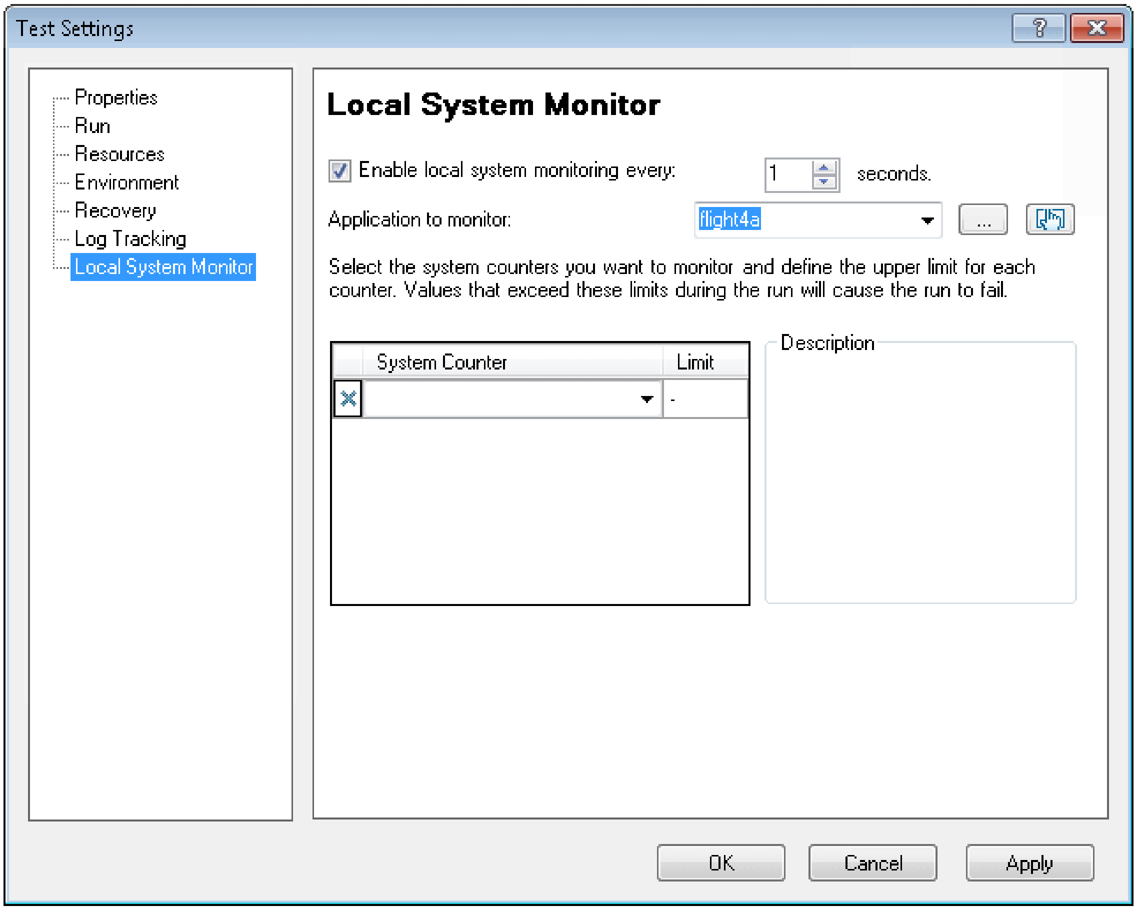
Local System Monitor Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)

Relevant for: GUI tests and components

This pane enables you to activate system monitoring, and define the system counters you want to track during a run session. These counters enable you to monitor the resources used by your application.

The following image shows the **Local System Monitor** pane in the Test Settings dialog box.

The panes available from the sidebar differ slightly in the Business Component Settings dialog box or an application area's Additional Settings pane.



Local System Monitor

☑ Enable local system monitoring every: 1 seconds.

Application to monitor: flight4a

Select the system counters you want to monitor and define the upper limit for each counter. Values that exceed these limits during the run will cause the run to fail.

System Counter	Limit
	-

Description

To access	Do one of the following: - For a test: With a test, action, or business component in focus, select **File > Settings > Local System Monitor** node. - For an application area: Open the application area and select **Additional Settings > Local System Monitor** in the sidebar.
Important information	- For components, this pane is read-only. The Local System Monitor data that is captured during a run session is displayed in the Run Results Viewer. For details, see the section on the the Run Results Viewer System Monitor Pane (described in the *HP Run Results Viewer User Guide*). - If more than one process with the same name runs during a run session, and you monitor a counter for that process (for example, you select to monitor a counter for the iexplorer.exe process, and more than one Internet Explorer browser is open on your desktop during the run session), the counter is sampled from the application that contains at least one test object from the test or component. If more than one application meets this criterion, only one application is monitored. - **For business process tests:** Local system monitoring runs only if the application area associated with the first component in the business process test is set to run local system monitoring, and only on the application defined in the Local System Monitor pane. This means that if another component that runs later in the business process test is associated with a different application area, local system monitor settings from that application area are ignored.
See also	"Local System Monitor" on page 585

User interface elements are described below:

UI Element	Description
Enable local system monitoring every: __ seconds	The frequency in seconds, by which the system counters for this application will be checked. Use the up and down arrows or enter a value in the edit box to change the number of seconds. **Minimum value:** One second.

UI Element	Description
Application to monitor	The application whose system counters you want to monitor. You can define the application in any of the following ways: • Enter the name of the application's executable file (without file extension) in the edit box. • Clicking the down arrow in the edit box, displays a list that contains the following: ▪ Applications previously run in UFT. ▪ Currently running applications. ▪ Applications currently specified in the Windows-based Applications tab of the Record and Run Settings dialog box. ▪ Applications currently specified in the "Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)". • Click the browse button ... and browse to the application's executable file. • Make sure that your application is currently running. Then click the pointing hand ☝ and point to the application on your desktop. **Note:** Sometimes a process is used only as a launcher that creates another process that provides the application functionality. Make sure to select the executable file that actually provides the application functionality.
System Counter	The system counter you want to track for the selected application. Click inside a cell and then click the down arrow. Select the counter from the list. Click the expand button ︾ when displayed to show more counters. You can monitor the process counters, which are accessible through the performance console (select **Start > Run** > and then enter Perfmon). For details these process counters, see the performance console's Help.
Limit	The upper limit of the counter selected in the **System Counter** column. If the selected counter exceeds this value during the run session, the run fails. The **Limit** value is optional. If you do not supply a value, the counter is tracked and the results are displayed in the Run Results Viewer.
☒	Removes the system counter definition from your test or application area.

UI Element	Description
Description	The description of the counter selected in the **System Counter** column, as provided by the performance console application.

Chapter 21: Setting GUI Testing Options Programmatically During the Run Session

Relevant for: GUI tests and scripted GUI components

This chapter includes:

Concepts

Setting GUI Testing Options Programmatically During the Run Session - Overview

Relevant for: GUI tests and scripted GUI components

You can use the **Setting** object to control how UFT runs tests by setting and retrieving testing options during a run session.

GUI testing options affect how you work with tests. For example, you can set the maximum time that UFT allows when loading a Web page, before determining that the URL address cannot be found.

You can set and retrieve the values of testing options during a run session using the Setting object in the Editor. For details on working in the Editor, see "Programming in GUI Testing Documents in the Editor" on page 994.

By retrieving and setting testing options using the Setting object, you can control how UFT runs a test.

You can also set many testing options using the Options dialog box (global testing options) and the Test Settings dialog box (test-specific settings). For details, see "UFT Global Options" on page 522 and "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.

This chapter describes some of the GUI testing options that can be used with the **Setting** object from within a test script. For detailed information on all the available methods and properties for the **Setting** object, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

> **Note:** You can also control UFT options as well as most other UFT operations using automation scripts. For details, see "UFT Automation Scripts" on page 1155 or the *HP Unified Functional Testing Automation Object Model Reference for GUI Testing* (**Help > HP UFT GUI Testing Advanced References > Automation Object Model Reference**).

Tasks

How to Set GUI Testing Options Programmatically During a Run Session

Relevant for: GUI tests and scripted GUI components

The following steps describe how you can set and retrieve the values of testing options during a run session using the **Setting** object in the Editor.

- "Set the value of a testing option from within the test script" below
- "Retrieve Testing Options" on the next page
- "Control the Test Run" on the next page
- "Add and Remove Run-Time Settings" on page 631

Set the value of a testing option from within the test script

To set the option, use the following syntax:

```
Setting (testing_option) = new_value
```

Some options are global and others affect only the current test. After you use a **Setting** object to set a testing option, the setting remains in effect until it is changed again or until the end of your current UFT session. You can also use the **Setting** object to change a setting for a specific part of a specific test. For details see "Control the Test Run" on the next page.

Some of the testing options that you can set using the **Setting** object are also available in the Options dialog box (global options) or the Test Settings dialog box (test specific settings). When you use the **Setting** object to set these options, the change is reflected in the relevant dialog box. Other test settings can be accessed using only one method, either the relevant dialog box or the **Setting** object. For more details, see "Setting GUI Testing Options Programmatically During the Run Session - Overview" on the previous page.

Example:

If you run the following statement with the Web Add-in loaded:

```
Setting("AutomaticLinkRun")=1
```

UFT disables automatically created checkpoints in the test. The setting remains in effect during your current UFT session until it is changed again, either with another Setting statement, or by clearing the **Ignore automatic checkpoints while running tests or components** check box in the Web Advanced pane (**Tools > Options > GUI Testing** tab **> Web > Advanced** node).

If you run the following statement:

```
Setting("WebTimeOut")=50000
```

UFT automatically changes the amount of time it waits for a Web page to load before running a test step to 50000 milliseconds. The setting remains in effect during your current UFT session until it is changed again, either with another Setting statement, or by setting the **Browser Navigation Timeout** option in the Web pane of the Test Settings dialog box (**File > Settings > Web** pane).

Note: Although the changes you make using the Setting object are reflected in the Options and Test Settings dialog boxes, these changes are not saved when you close UFT, unless you make other changes in the same dialog box manually and click **Apply** or **OK** (which saves all current settings in that dialog box).

Retrieve Testing Options

You can use the **Setting** object to retrieve the current value of a testing option.

To store the value in a variable, use the syntax:

```
new_var = Setting ( testing_option )
```

To display the value in a message box, use the syntax:

```
MsgBox (Setting (testing_option) )
```

For example:

```
LinkCheckSet = Setting("AutomaticLinkRun")
```

assigns the current value of the AutomaticLinkRun setting to the user-defined variable LinkCheckSet.

Control the Test Run

You can use the retrieve and set capabilities of the **Setting** object together to control a run session without changing global settings. For example, if you want to change the **DefaultTimeOut** testing option to 5 seconds for objects on one Web page only, insert the following statement after the Web page opens in your test script:

```
‘Keep the original value of the DefaultTimeOut testing option
old_delay = Setting ("DefaultTimeOut")
‘Set temporary value for the DefaultTimeOut testing option
Setting("DefaultTimeOut")= 5000
```

To return the **DefaultTimeOut** testing option to its original value at the end of the Web page, insert the following statement immediately before linking to the next page in the script:

```
`Change the DefaultTimeOut testing option back to its
`original value.

Setting("DefaultTimeOut")=old_delay
```

Add and Remove Run-Time Settings

You can add, modify, and remove custom run-time settings. These settings are applicable during the run session only.

To add a new run-time setting, use the syntax:

> **Setting.Add "***testing_option***", "***value***"**

For example, you could create a setting that indicates the name of the current tester and displays the name in a message box.

```
‘Change the DefaultTimeOut testing option back to its original
‘value.
Setting("DefaultTimeOut")=old_delay
```

> **Tip:** When using a **Setting.Add** statement, an error occurs if you try to add an existing setting option. To avoid this error you should use a **Setting.Exists** statement first. For details about all the **Setting** methods, see the *HP UFT Object Model Reference for GUI Testing*.

To modify a run-time setting that has already been initialized, use the same syntax you use for setting any standard setting option:

> **Setting (** *testing_option* **) =** *new_value*

For example:

```
Setting("Tester Name")="Alice Wonderland"
```

To delete a custom run-time setting, use the following syntax:

> **Setting.Remove (** *testing_option* **)**

For example:

```
Setting.Remove ("Tester Name")
```

Tip: When using a **Setting.Remove** statement, an error occurs if you try to remove a setting option that does not exist. To avoid this error you should use a **Setting.Exists** statement first. For more details about all the **Setting** methods, see the *HP UFT Object Model Reference for GUI Testing*.

Part 4: UFT Running and Debugging Operations

Chapter 22: Running Tests and Components

Relevant for: GUI tests and components and API testing

This chapter includes:

Concepts

Running GUI Tests and Components - Overview

Relevant for: GUI tests and components

When you run a test or component, UFT performs the steps it contains. If you have defined test or component parameters, UFT prompts you to enter values for them. When the run session is complete, UFT displays a report detailing the results. For details on viewing run results, see the *HP Run Results Viewer User Guide*.

For details about running business process tests, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

Run Preferences

UFT always runs a test or component from the first step, unless you specify otherwise. You can:

- Run the entire test or component from the beginning.
- Run only a part of a test using the following options:
  - **Run from Action**
  - **Run to Action**
- Run only a part of a test or component. These features are useful if you want to check a specific section of the test or component, or to confirm that a certain part of your test or component runs smoothly, without running it from the beginning or to the end. The **Run to Step** and the **Run to Action** options are also useful if you want to open your application to a specific location to add steps or debug your test or component. Use the following options:
  - **Run from Step**
  - **Run to Step**
- Debug a section of a test or component using the **Debug from step** or the **Debug from Action** option. (Make sure that the application is open to the relevant location before using this option.) For details, see "Running to a Step and Debugging from a Step " on page 678.
- Update your test or component to change the test object descriptions.

  For tests: You can also change the expected checkpoint values, and/or the Active Screen images and values. For details, see "Maintaining and Updating GUI Tests or Components " on page 1078.
- Run tests and components on objects with dynamic descriptions. For details, see "Managing Test Objects in Object Repositories" on page 1198.

- **For tests:**
  - Run an iteration of a single action using the **Run Current Action** option. Even though this option runs only one iteration, if the action contains nested actions, UFT runs the nested actions for the defined number of iterations.
  - Run only one iteration of your entire test by selecting **Run one iteration only** from the Run pane in the Test Settings dialog box. For details, see "Run Pane (Test Settings Dialog Box)" on page 601.
  - Designate certain steps as **optional**, to enable UFT to bypass them instead of aborting the run if these steps do not succeed. For details, see "Default Optional Steps" on page 650.
  - Set up a batch of tests and run them sequentially, using the UFT Test Batch Runner. For details, see "Test Batch Runner Window" on page 656.

For tests: Global Data Table Parameters

If your test contains a global Data Table parameter, UFT runs the test once for each row in the Data pane. If your test contains a Data Table parameter for the current action data sheet, UFT runs the action once for each row of data in that action data sheet. You can also specify whether to run the first iteration or all iterations, for the entire test or for a specific action in the test; or to run the iterations for a specified range of data sets. For details on test iterations, see "Run Pane (Test Settings Dialog Box)" on page 601. For details on Data Table parameters see, "Data Table Parameters" on page 1533.

Running API Tests - Overview

Relevant for: API testing

You can run tests directly from the UFT interface or from the command line. The test run enables you to determine if the activities are functional and if they are performing as expected.

Before running the test, you can select a location in which to store the results, and set input parameter values specifically for this test run. For details, see "Run Dialog Box" on page 651.

For details about the command line, see "Command Line Syntax for Running API Tests" on page 659.

The Run Results Viewer provides a detailed report about each step and its checkpoints.

When you run custom code from the user interface, you can also use breakpoints and other tools to debug your code.

Using the Options dialog box, you can set the run configuration—**Debug** or **Release**. The **Release** mode is more efficient, and is recommended for Load Testing. For details, see "General Pane (Options Dialog Box > API Testing Tab)" on page 565.

When a test step fails, by default, UFT stops the test. You can instruct UFT to continue the test if you set the **Stop test if checkpoint fails** option to NO in the Test Settings tab. For details, see "Test Settings Tab (Properties Pane - API Testing)" on page 429.

For Web Service steps, you can validate the SOAP response against a schema or WS-I standards. For details, see "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

Optional Steps

Relevant for: GUI tests and scripted GUI components

An **optional** step is a step that is not necessarily required to successfully complete a run session. For example, suppose that when creating a test or component, you add login steps because the application you are testing prompts you to enter a user name and password in a login window. Suppose, too, that this particular application remembers user login details, so that you do not need to log in every time you open the application. During a run session, the application does not prompt you to enter your user name and password because it retained the information that was previously entered. In this case, the steps that you added for entering the login information are not required and should, therefore, be marked optional.

During a run session, if the object of an optional step does not exist in the application, UFT bypasses this step and continues to run the test or component. When the run session ends, a message is displayed for the step indicating that the step was not performed, but the step does not cause the run to fail.

However, if, during a run session, UFT cannot find the object from the optional step in the object repository (for example, if the object name was modified in the test or component but not in the object repository, or if the object was removed from the object repository), an error message is displayed listing the required object, and the run fails.

During a recording session, UFT automatically marks steps that open certain dialog boxes as optional. (For a list of these dialog boxes, see "Default Optional Steps" on page 650.)

You can also manually designate steps as optional. For example, you can add conditional statements or use recovery scenarios to automatically click a button, press ENTER, or enter login information in a step. For details, see "Conditional Statements" on page 970, "How to Insert Conditional Statements from the Keyword View" on page 930, and "Recovery Scenarios for GUI Testing" on page 1111.

For details, see:

- "How to Set Optional Steps"
- "Default Optional Steps"

Test Batch Runner Overview

Relevant for: GUI testing and API testing

Test Batch Runner enables you to run tests in a collective, successive test run. Tests are run individually but sequentially in a single session.

You can use Test Batch Runner only on tests stored on the file system. You cannot use Test Batch Runner for tests stored in ALM.

Using Test Batch Runner, you create a list of tests and save the list as a batch .mtb file, so that you can run the same batch of tests again at another time. You can choose to include or exclude a test in your batch list from running during a particular batch run without affecting the other tests in the batch.

You can add tests individually to Test Batch Runner by navigating to the folder in which the test is located. Test batches can also be added to another test batch by adding an .mtb test batch file to a new test batch. When Test Batch Runner opens, it checks to make sure that all tests within a test batch exist.

When running the test batch, the Output pane allows you view the results of the test run in run time. During the test batch run, the Output pane displays the test's path in the file system, the progress of the test, as well as any errors that occur during the run.

Following the test batch run, the results are saved to a run results file including whether the test passed or failed and errors in running the test. The **Report** column of the **Tests** pane displays a link to the run results file.

If you do not want to run the test batch through the Test Batch Runner interface, you can run the Test Batch Runner from the Windows Command Line. You provide the location of a .mtb file, a test folder, or a directory of tests as a command argument and Test Batch Runner runs the test batch and displays the run results.

Tip: Using the command line options of the Test Batch Runner, you can include UFT tests in as part of build runs in a continuous integration system.

Tasks

For details about running business process tests, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

How to Run a GUI Test, Business Process Test, or Component

Relevant for: GUI tests and scripted GUI components

This task describes the various ways in which you can run a test or component.

This task includes the following steps:

- "Prerequisites" below
- "Run an entire test or component" below
- "Run to a selected step or action" on the next page
- "Run a single action, test, or a component from a selected step" on the next page
- "Interrupt a run session" on page 641
- "Results" on page 642

Prerequisites

1. Ensure that any required UFT add-ins are loaded. For details, see "How to Start UFT" on page 114.
2. Open the test or component you want to run. For details, see "Open/New <Document>/<Resource> Dialog Box" on page 156

Run an entire test or component

1. Open the Run dialog box in one of the following ways:
   - Click the **Run** button.
   - Select **Run > Run**.

   The Run dialog box opens.

2. In the Run dialog box, choose where to save the run session results, and define any input parameters you want to use, as described in "Run Dialog Box" on page 651, and "Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)" on page 655.

   > **Note:** (For tests) When running part of a test within the scope of an action, you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box. For details, see "Parameters Tab (Action Properties Dialog Box)" on page 903.

3. Click **OK**. The Run dialog box closes and the run session starts.

   > **Note: For GUI testing:** When running a test with external resource files (like function libraries, data tables, recovery scenarios,etc.) that are saved in ALM, the resource files are not refreshed for each test run. As a result, any changes made to these external resource files during the current session are not reflected in a test run until you close and reload test and its resource files.

Run to a selected step or action

1. Do one of the following:

   - **For tests:**

     - Select **Run > Run to Step**.

     - Right-click a step and select **Run to Step**.

     - Right-click an action in the canvas and select **Run to Action**.

   - **For components:** Select **Run > Run to Step**.

2. In the Run dialog box, choose where to save the run session results, and define any input parameters you want to use, as described in "Run Dialog Box" on page 651, and "Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)" on page 655.

   > **Note:** (For tests) When running part of a test within the scope of an action, you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box. For details, see "Parameters Tab (Action Properties Dialog Box)" on page 903.

   The run starts at the beginning of the test or component and pauses at the selected step.

Run a single action, test, or a component from a selected step

1. Make sure your application is in a state matching the step or action you want to run.

2. Select the step or action where you want to start running the test or component.

   Make sure that the step or action you choose is not dependent on previous steps, such as a retrieved value or a parameter defined in a previous step.

   For tests, do one of the following:

   - In the canvas, select the action.
   - In the Keyword View, highlight a step or action row.
   - In the Editor, place your cursor in a line of VBScript.

3. Do one of the following:

   - **For tests:**
     - Select **Run > Run from Step.**
     - Select **Run > Run Current Action.**
     - Right-click the step and select **Run from Step**.
     - Right-click the action in the canvas and select **Run from Action**.
   - **For components:** Select **Run > Run from Step.**

     The Run dialog box opens.

4. In the Run dialog box, choose where to save the run session results, and define any input parameters you want to use, as described in "Run Dialog Box" on page 651, and "Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)" on page 655.

   > **Note:** (For tests) When running part of a test within the scope of an action, you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box. For details, see "Parameters Tab (Action Properties Dialog Box)" on page 903.

Interrupt a run session

Do one of the following:

- Click the **Pause** button ⏸ in the toolbar or select **Run > Pause**. The run pauses. To resume running a paused run session, click the **Run** button or select **Run > Run**. (Before using the **Pause** button, select **Run test in debug mode** in the **Options > API Testing** tab > **General** node.)

- Click the **Stop** button, select **Run > Stop**, or press **F4**.

- Perform a file operation (for example, open a different test or component or create a new test or component).

Results

By default, when the run session ends, the Run Results Viewer opens. For details, see the Run Results Viewer Overview (described in the *HP Run Results Viewer User Guide*). If you run part of a test or component, the Run Results summary displays a note indicating that the test or component was run using the **Run from Step** or **Run Current Action** option.

> **Note:** If you cleared the **View results when run session ends** check box in the **Run Sessions** pane of the Options dialog box (**Tools > Options > General** tab **> Run Sessions** node), the Run Results Viewer does not open at the end of the run session. For details, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 526.

You can also optionally automatically upload your run results to ALM if you are running a test from ALM. This option is set in ALM as a site parameter for your project. For details, see the *HP Application Lifecycle Management Administrator Guide*.

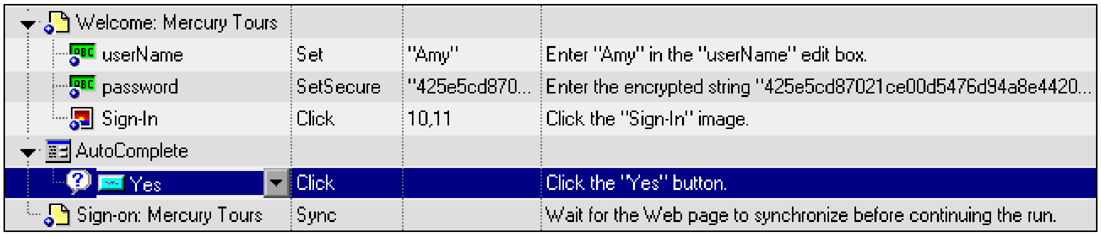
How to Set Optional Steps

Relevant for: GUI tests and scripted GUI components

This task describes how to set an optional step.

Do one of the following:

- In the Keyword View, right-click the step and select **Optional Step**. The Optional Step icon is added next to the selected step.



Welcome: Mercury Tours			
userName	Set	"Amy"	Enter "Amy" in the "userName" edit box.
password	SetSecure	"425e5cd870...	Enter the encrypted string "425e5cd87021ce00d5476d94a8e4420...
Sign-In	Click	10,11	Click the "Sign-In" image.
AutoComplete			
Yes	Click		Click the "Yes" button.
Sign-on: Mercury Tours	Sync		Wait for the Web page to synchronize before continuing the run.

- In the Editor, add OptionalStep to the beginning of the VBScript statement. For example:

```
OptionalStep.Browser("Browser").Dialog("AutoComplete").WinButton("Yes").Click
```

For details on working in the Editor, see "Programming in GUI Testing Documents in the Editor" on page 994.

How to Run an API Test

Relevant for: API tests

This task describes how to run a test through the user interface and view its results.

For details on running a test through the command line, see "Command Line Syntax for Running API Tests" on page 659.

This task includes the following steps:

- "Add new tests to a solution - optional" below
- "Set values for the test variables - optional" on the next page
- "Configure checkpoints" on the next page
- "Save the test in ALM - optional" on the next page
- "Add external references - optional" on page 645
- "Set the number of iterations" on page 645
- "Select a location for the run results - optional" on page 645
- "Set runtime property values - optional" on page 645
- "Validate the structure of the SOAP response " on page 645
- "Run the test" on page 645
- "Debug the test - optional " on page 645
- "Analyze the Results" on page 646

1. Add new tests to a solution - optional

To add more tests to the solution, select **File > Add > New Test** or **File > Add > Existing Test.** To add tests to an ALM repository, first connect to the ALM server. For details, see "ALM Connection Dialog Box" on page 737.

UFT adds the test to the Solution Explorer pane.

For user interface details, see the "Add Test/Component to Solution Dialog Box" on page 146 or "Add <Existing Document> to Solution Dialog Box" on page 145

> **Note:** If you are running an API test that is stored in ALM and calls a GUI test, you must either ensure that UFT is open with a solution open, or you must close UFT and allow ALM to open it. The test will not run if UFT is open with no solution.

2. **Set values for the test variables - optional**

   Select the test properties or user/system variables. For details, see "How to Define API Test Properties or User/System Variables" on page 141.

3. **Configure checkpoints**

   Checkpoints let you validate the results. For details, see "Checkpoint Validation for Test Steps" on page 1612.

   To add checkpoints, do the following:

   a. Click the **Input/Checkpoints** tab . By default, the pane displays the **XML** tab. Provide an expected value for each row in one of the following ways:

      - Manually
      - Select Link Source dialog box .
      - **Load from replay** button .

   b. Select a comparison operator, such as =, >, or Regex. These may differ depending on the data type.

   c. Select the **Validate** check boxes in the rows of the properties that you want to validate. Clear the check boxes for the properties you are not validating.

   d. Set checkpoints for array elements. For details, see "How to Set Array Checkpoints for Test Steps" on page 1615:

   e. To validate against an XPath expression, click the **XPath** tab and provide the expression. For details, see "How to Set XPath Checkpoints for Test Steps" on page 1617:

   f. To halt the test run if the response does not return the expected value select **Stop test if checkpoint fails**.

      > **Tip:** You can customize and override the checkpoint settings using an event handler. For details, see "Set the value of a checkpoint" on page 1970.

      For user interface details, see the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

      For details on how to work with array checkpoints, see "How to Set Array Checkpoints for Test Steps" on page 1615.

4. **Save the test in ALM - optional**

   To save results in ALM, you need to first save the actual test in ALM. Connect to ALM and then select **File > Save as.**

5. Add external references - optional

You can add one or more references to your test.

a. Open the Solution Explorer pane.

b. Expand the test's tree and select the **References** node.

c. Select **Add Reference** from the context menu.

d. Specify an external .dll file as a reference.

For details, see the "Add Reference Dialog Box" on page 485.

6. Set the number of iterations

Click in the **Test Flow** or **Loop** box and select the **Input/Checkpoints** tab . Set the number of iterations in the Properties pane. For details, see "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

7. Select a location for the run results - optional

Tip: To run a test bypassing the Run dialog box, select **Run > Run Now**.

a. Click the **Run** button on the toolbar and select the **Results Location** tab.

b. For tests saved on the File System, select a location in which to store the run results—**New run results folder** or **Temporary run results folder**. Choose the latter if you do not expect to use and analyze the results. For details, see "Run Dialog Box: Results Location Tab (For GUI and API Tests Stored on File System)" on page 653.

c. For tests saved in ALM, specify a run name. If relevant, select a **Test Set** and **Instance**.

8. Set runtime property values - optional

In the Run dialog box, click the **Input Parameters** tab and set the values that you want the run session to use for the listed properties.

9. Validate the structure of the SOAP response

For Web Service steps, you can instruct UFT to validate the SOAP response against the schema or WS-I standards. Set the desired validations in the bottom of the Checkpoints area. For details, see the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

10. Run the test

Click the **OK** button in the Run dialog box.

11. Debug the test - optional

By default, UFT runs tests in release mode—not debugging mode. This allows the test to run much quicker. To run the test in debugging mode, which also allows you to pause the test during its run, you need to enable it in the Options dialog box. For details, see "General Pane (Options Dialog Box > API Testing Tab)" on page 565.

In debugging mode, select **Run > Pause** to stop the test run at any point. Use the debug panes to solve any runtime issues. For user interface details, see the or "Debug Panes" on page 271.

12. **Analyze the Results**

    View the results in the Run Results Viewer. The Run Results Viewer opens automatically after a test run.

    - A green check mark indicates success.

    - A red **X** mark indicates a failure in one of the steps. Expand the nodes of the report to view details about each step and checkpoint.

    To view the report at a later stage, select **View > Last Run Results**.

    To prevent the Results viewer from opening, select **Tools > Options > General** tab **> General** node**.** Clear the **View results when run session ends** option.

    You can also optionally automatically upload your run results to ALM if you are running a test from ALM. This option is set in ALM as a site parameter for your project. For details, see the *HP Application Lifecycle Management Administrator Guide*.

How to Create and Run a Test Batch

Relevant for: GUI tests and components and API testing

This task explains how to create and run a test batch using Test Batch Runner.

This task includes the following steps:

- "Open Test Batch Runner" on the next page

- "Access the test batch file" on the next page

- "Add batches or tests" on the next page

- "Select the tests to be part of the test batch run" on page 648

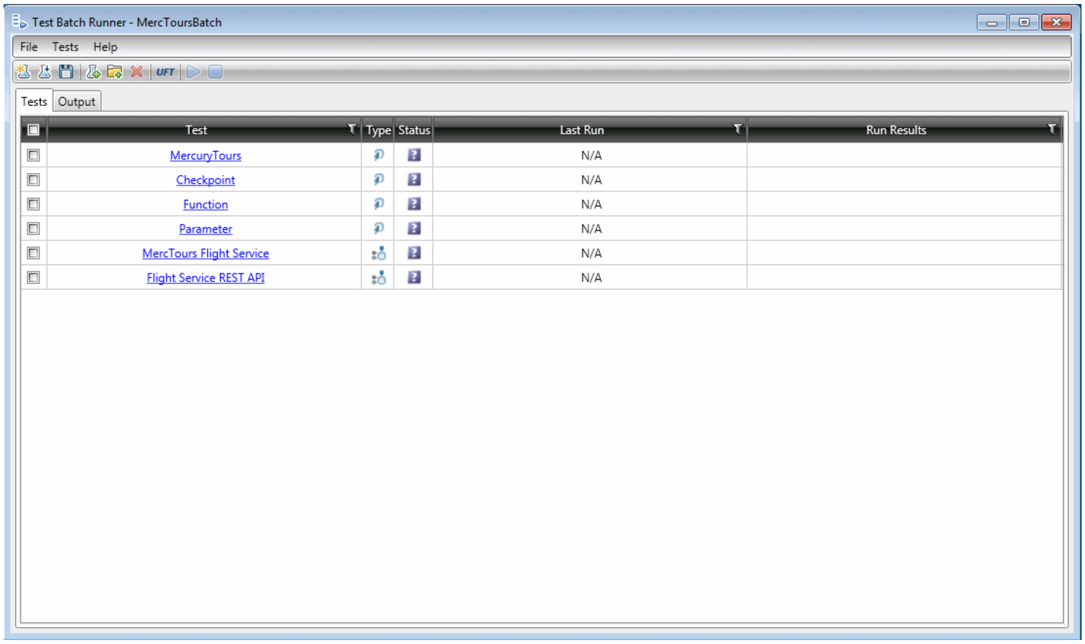
- "Save the test batch" on page 648

- "Run the test batch" on page 648

- "View the test batch run results" on page 648

1. ## Open Test Batch Runner

   Select **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Test Batch Runner**. This opens a separate window for the Test Batch Runner program.



☐	Test	Type	Status	Last Run	Run Results
☐	MercuryTours			N/A	
☐	Checkpoint			N/A	
☐	Function			N/A	
☐	Parameter			N/A	
☐	MercTours Flight Service			N/A	
☐	Flight Service REST API			N/A	

Note:

- You do not need to have UFT open to use Test Batch Runner.
- For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

2. ## Access the test batch file

   - To create a new test batch file, select **File > New** or click the **New** batch file button .
   - To open an existing batch file, select **File > Open** or click the **Open** batch file button . In the Open dialog box, navigate to the folder in which the batch file is found and click **Open.** The tests from the opened batch file are added to the Test Batch Runner main window.

3. ## Add batches or tests

   - To add a test batch file (**.mtb**), select **File > Add** or click the **Add** button . Navigate to the folder in which the batch file is saved.

- To add individual tests, select **Tests > Add** or click the **Add** button . In the **Browse For Folder** dialog box, select the folder in which your tests are located. All the tests from the selected folder are added to the **Tests** pane in the main Test Batch Runner window.

  > **Note:** When adding tests through the **Tests > Add** menu command, you must select all the tests from the target folder. If you do not want to run all the tests in the target folder, select the check boxes next to the tests you want to run before you run the test batch.

4. Select the tests to be part of the test batch run

Select the checkboxes □ for the tests that you want to include in the test batch run.

5. Save the test batch

Select **File > Save** or click the **Save** button in the toolbar to save the test batch with the included test. Give your batch file a meaningful name and assign it to a place in your directory.

6. Run the test batch

Click the **Run** button ▶ to run the test batch. The Output pane provides run log details of the batch run while the batch is running.

7. View the test batch run results

In the Tests pane, click the results link for a specific test in the **Run Results** column. This opens the results for that test in the Run Results Viewer. For details on the Run Results Viewer, click F1 within the viewer window.

How to Run a Test Batch Using the Windows Command Line

Relevant for: GUI tests and components and API testing

This task describes how to run a test using the Windows Command Line.

This task includes the following steps:

- "Open the Windows Command Line window" on the next page
- "Provide the source folder for the batch file or tests" on the next page
- "Run the test batch" on the next page
- "View the test batch run results (API testing only)" on the next page

1. **Open the Windows Command Line window**

   Run cmd.exe to open the Command Line window. (For example, from the Windows Run dialog box.)

2. **Provide the source folder for the batch file or tests**

   In the Command Line window, enter UFTBatchRunnerCMD.exe and the **source** switch followed by the test batch file (.mtb) or folder containing the test.

   For example, your command line might contain text like this:

   ```
   UFTBatchRunnerCMD.exe -source "C:\users\MySample.mtb"
   UFTBatchRunnerCMD.exe -source "C:\users\APITest1"
   ```

3. **Run the test batch**

   After entering the Test Batch Runner command and the location of the folder containing your tests, press ENTER. Test Batch Runner runs the test batch. For API tests, the test log is displayed in the command window.

4. **View the test batch run results (API testing only)**

   When the test batch run is complete:

   a. Open the Run Results Viewer from the **Start** menu. The Open Run Results dialog box opens.

      > **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

   b. In the Open Run Results dialog box, select the **Results XML file** option.

   c. Navigate to the Results.xml file under the test's **Report** folder and click **Open**.

      Each test has its own results. For details, see the *HP Run Results Viewer User Guide*.

Reference

Default Optional Steps

Relevant for: GUI tests and scripted GUI components

By default, UFT considers steps that open the following dialog boxes or message boxes as optional steps:

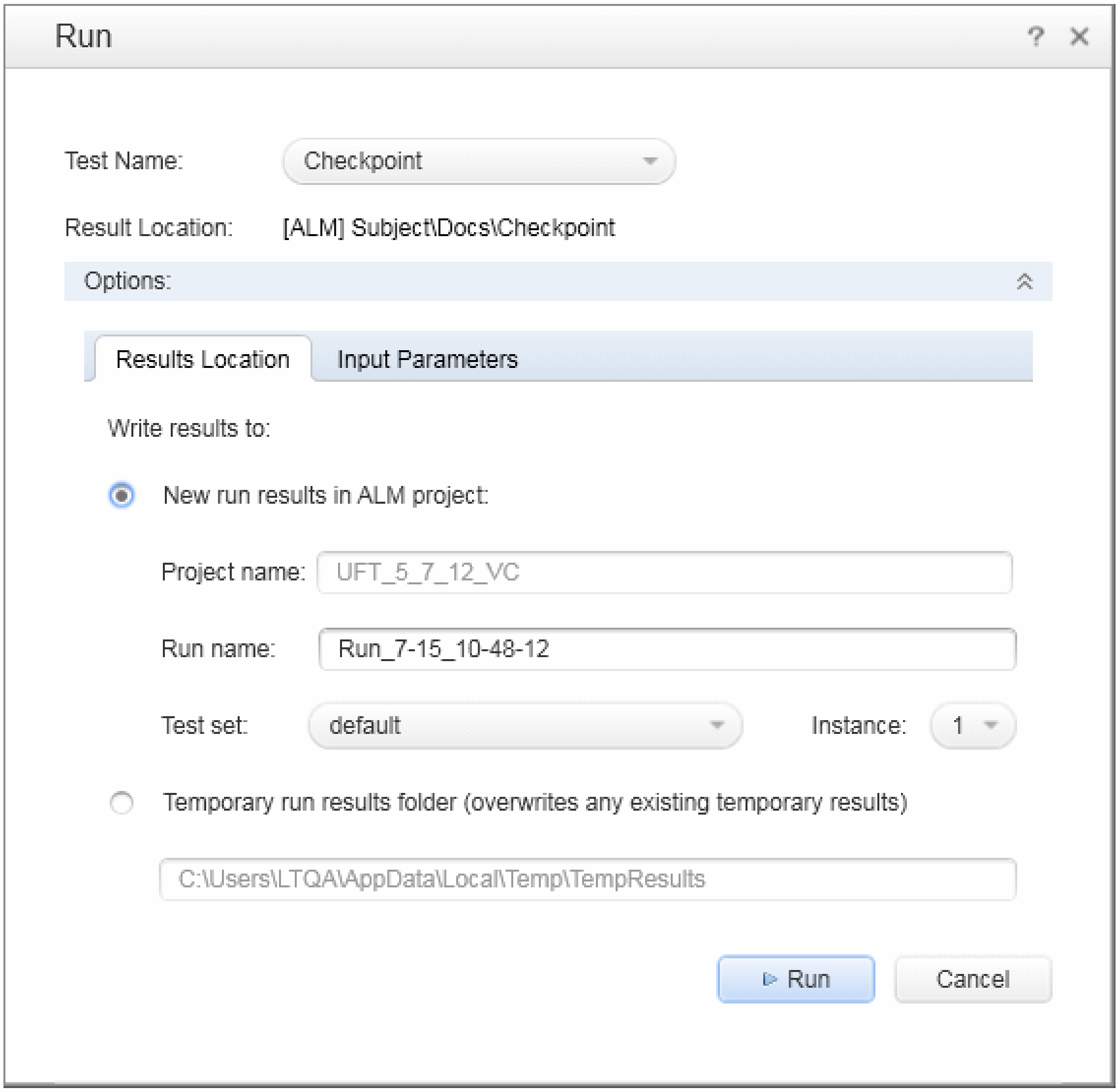
Dialog Box / Message Box Title Bar
AutoComplete
File Download
Internet Explorer
Enter Network Password
Error
Security Alert
Security Information
Security Warning
Username and Password Required

For an overview and task details, see "Optional Steps" on page 637 and "How to Set Optional Steps" on page 642.

Run Dialog Box

Relevant for: GUI testing, API testing, and Business Process Testing

This dialog box lets you prepare for the test run by selecting the test to run and providing a location for the results.



Run

Test Name: Checkpoint

Result Location: [ALM] Subject\Docs\Checkpoint

Options:

Results Location | Input Parameters

Write results to:

● New run results in ALM project:

Project name: UFT_5_7_12_VC

Run name: Run_7-15_10-48-12

Test set: default Instance: 1

○ Temporary run results folder (overwrites any existing temporary results)

C:\Users\LTQA\AppData\Local\Temp\TempResults

To access	1. Do one of the following: ▪ Ensure that a test or component is in focus in the document pane. ▪ In the Solution Explorer, select a test or component node. 2. Select one of the following: ▪ Click the **Run** button or select **Run > Run**. ▪ For GUI tests: ○ Click the **Run** button and select **Maintenance Run Mode** or select **Run > Maintenance Run Mode**. ○ Click the **Run** button and select **Update Run Mode** or select **Run > Update Run Mode.**
Important information (for GUI and API tests)	As a test runs, each step is highlighted in the Keyword View (GUI Tests only). • When the test stops running, the Run Results Viewer opens unless you have cleared the **View results when test run ends** check box in the **Run Sessions** pane of the Options dialog box (**Tools > Options > General** tab **> Run Sessions** node). For details, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 526. • You can also optionally automatically upload your run results to ALM if you are running a test from ALM. This option is set in ALM as a site parameter for your project. For details, see the *HP Application Lifecycle Management Administrator Guide*. • You can report defects to an ALM project either automatically as they occur, or manually directly from the Run Results Viewer. For details, see the online help in the Run Results Viewer. • When running API tests, you can bypass this dialog box using the **Run > Run Now** menu item. • **For GUI testing:** When running a test with external resource files (like function libraries, data tables, recovery scenarios,etc.) that are saved in ALM, the resource files are not refreshed for each test run. As a result, any changes made to these external resource files during the current session are not reflected in a test run until you close and reload test and its resource files.

Important Information (for business process tests and flows)	When you run business process tests and flows in UFT, you can only select the document you are running, if you have multiple documents open. You cannot modify the results location, nor can you define any other options. **Note:** For improved performance when running business process tests or flows, UFT creates and runs a hosting test, named **Test Runtime**. The **Test Runtime** test is recreated each time the test or flow runs, and is not saved with the run.
Relevant tasks	• "How to Run a GUI Test, Business Process Test, or Component" on page 639 • "How to Use Maintenance Run Mode to Update Your Test or Component When Your Application Changes " on page 1085 • "Running to a Step and Debugging from a Step " on page 678 • "How to Run an API Test" on page 643 • "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

UI Elements	Description
Test Name	The test or component to run. You can select any open item from the drop-down list.
Results Location (for GUI, API, and business process tests)	The target location for the results. For GUI and API tests, you can specify this location in the **Results Location** tab. For business process tests, results are always saved in the temporary run results folder.
Options (for GUI and API tests) ⌄ ⌃	Expands or collapses the dialog box to show the **Results Location** and **Input Parameters** tabs.
Run	Begins the run session.

Run Dialog Box: Results Location Tab (For GUI and API Tests Stored on File System)

This tab enables you to specify where to save run results when running tests stored on the file system.

User interface elements are described below:

UI Elements	Description
New run results folder	An area allowing you to indicate the results location. The area displays the default path and folder name in which the results are saved. A new folder is created for each run. By default, results are stored in the following locations: • **For tests:** The test folder • **For components:** An ALM cache folder on your computer
Temporary run results folder	Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder. **Note:** • The path in the text box of the **Temporary run results folder** option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts. • UFT stores temporary results in %TMP%\TempResults (which is usually <System Drive>\Documents and Settings\<user name>\Local Settings\Temp\TempResults).

Run Dialog Box: Results Location Tab (For GUI and APITests Stored in ALM)

This tab enables you to specify where to save run results when running tests stored in ALM.

User interface elements are described below:

UI Elements	Description
New run results in ALM project	• **Project name.** Displays the ALM project to which you are currently connected. • **Run name.** The name of the run. You can accept the automatically generated name or enter a different one. • **Test set.** A group of tests selected to achieve specific testing goals. For example, you can create a test set that tests the user interface of the application or the application's performance under stress. (You define test sets when working in the ALM test run mode. For details, see your ALM documentation.) • **Instance.** The instance of the test in the test set. If there is more than one instance, select the instance of the test for which you want to save the results.
Temporary run results folder	Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder. **Note:** • The path in the text box of the **Temporary run results folder** option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts. • UFT stores temporary results in %TMP%\TempResults (which is usually <System Drive>\Documents and Settings\<user name>\Local Settings\Temp\TempResults)

Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)

This tab enables you to specify to specify the run-time values of input parameters to be used during the run session.

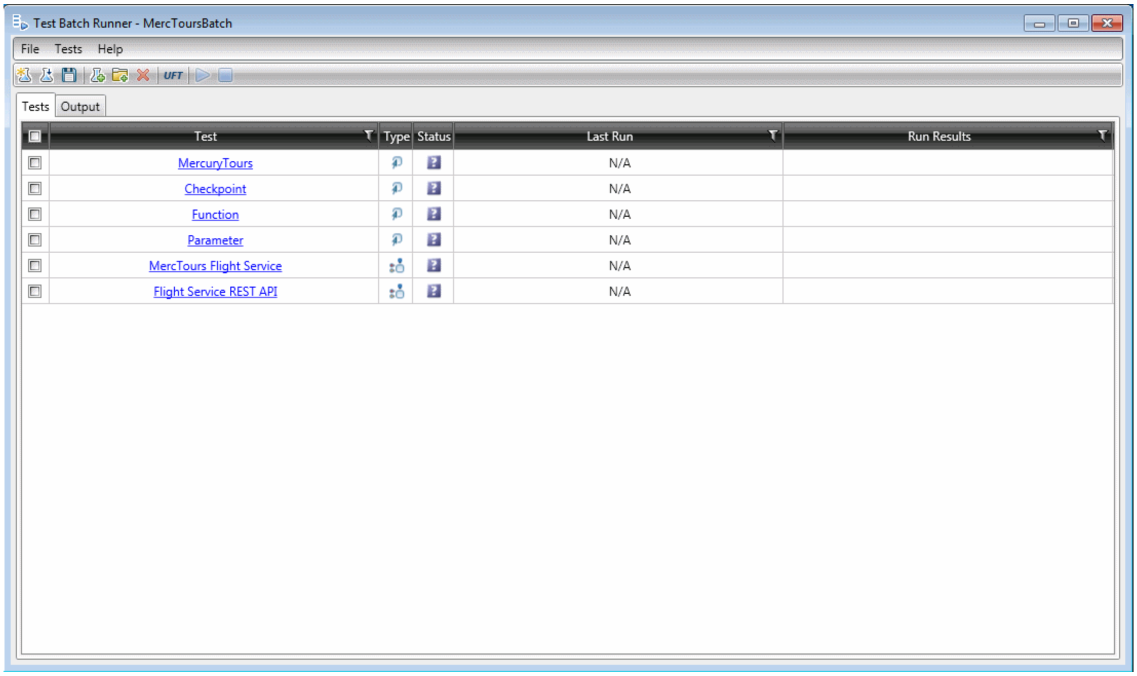
User interface elements are described below:

UI Elements	Description
Input parameters	The input parameters that were defined for the test or component. **To set the value of a parameter to be used during the run session:** Click in the **Value** field for the specific parameter and enter the value, or select a value from the list. If you do not enter a value, it uses the default value from the Test Settings or Business Component Settings dialog box during the run session.

Test Batch Runner Window

Relevant for: GUI tests and API tests

The Test Batch Runner enables you to create and maintain test batch files and add tests to a test batch file.



□	Test	Type	Status	Last Run	Run Results
□	MercuryTours		?	N/A	
□	Checkpoint		?	N/A	
□	Function		?	N/A	
□	Parameter		?	N/A	
□	MercTours Flight Service		?	N/A	
□	Flight Service REST API		?	N/A	

To access	Select **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Test Batch Runner.** **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.
Important information	- You can use Test Batch Runner without having UFT open. - Test Batch Runner cannot include tests created in QuickTest 9.2 or earlier or Service Test 10.00 or earlier. - The panes within this window can be moved or pinned. Click on a pane and drag it to move it to your desired location, or right-click on the title of the pane to change the options for its display. For more information on how UFT windows can be displayed, see "How to Customize the UFT Window " on page 184.
Relevant tasks	"How to Create and Run a Test Batch" on page 646
See also	"Test Batch Runner Menu Commands" on page 658

Tests Pane

The user interface elements below describe the columns in the Tests pane.

UI Element	Description
□	Checkboxes that enable you to select the tests to include in the current test batch run.
Test	The name of the tests or test batches included in the test batch.
▼	Opens the filter window that enables you to group tests and results included in the batch file. You can filter the tests by a number of criteria using the drop-down menus at the bottom of the filter window. You can use multiple filters by choosing the combination argument from the middle drop-down menu. **Note:** This filter can be used for both the **Test**, **Last Run**, and **Run Results** columns.
Type	Displays the type of test: Service Test (ST)/UFT API) or QuickTest (QTP)/UFT GUI.
Status	Displays the run status of the test: • ? **Unknown.** The test has not been run or its status is unknown. • ✓ **Passed.** • ✗ **Failed.** • ◌ **Running.** • ! **Error.** The test encountered an error when running the test, such as Test Batch Runner not being able to find the test or a test missing an testing object. **Note:** If the status is displayed as **Error**, hover over the error icon to see a short description of the error.
Run Results	The link to the folder in which the results for the test is saved.

Output Pane

The Output pane displays the run long of the test batch. This includes information on the test run:

- the test currently running
- the step that is currently running within a test

- errors encountered during the test run
- the location of the run results.

> **Note:** This area will remain blank unless there is a test batch running or a batch that has completed running.

Test Batch Runner Menu Commands

Relevant for: GUI testing and API testing

You manage your test batch and individual tests the **File** and **Tests** menu commands.

File Menu Commands

File menu commands are used to create, update, and maintain test batch (.mtb) files.

	Command	Shortcut Key	Function
	New	**CTRL+N**	Creates a new batch file (.mtb).
	Open	**CTRL+O**	Opens an existing batch file.
	Add	**INSERT**	Adds a batch file to the current batch.
	Save	**CTRL+S**	Saves a new batch file.
	Save As		Enables you to save the batch file under another name.
	Exit	**ALT+F4**	Closes Test Batch Runner.

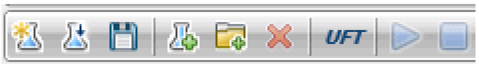
Tests Menu Commands

Test menu commands are used to add and maintain individual tests within the batch file.

	Command	Shortcut key	Function
	Add		Adds individual tests (not batch files) to the current batch. **Note:** When using this command, you must add all tests from the selected folder to the batch file.
	Remove	**DEL**	Removes a test from the batch file.
	Run	**F5**	Runs the selected tests in the batch file.
	Stop		Stops the test batch run.

Test Batch Runner Toolbar

Many of the menu commands are also available by default from the toolbar, shown below:



In addition, the UFT button is not present in the **File** or **Tests** menus. It is described below:

UI Element	Description
UFT	**Use HP Unified Functional Testing license.** Instructs Test Batch Runner to use a Unified Functional Testing license during a batch run. Use this button only if your batch contains both Service Test/UFT API tests and QuickTest/UFT GUI tests. **How does it work?** When you click **Run** ▷ (or select **Tests > Run**), Test Batch Runner instructs UFT to use an HP Unified Functional Testing license. If it cannot use the license type, an error message opens informing you that if your run a test batch with both Service Test/UFT API tests and QuickTest/UFT API tests, they will fail. **Note:** If you click this button and UFT begins using an HP Unified Functional Testing license, UFT will continue to use this license for the duration of the batch run session. To clear this license, you must close the Test Batch Runner.

Command Line Syntax for Running API Tests

Relevant for: API testing only

You can also run API tests using the ServiceTestExecuter.exe application, located in the product's bin directory.

The following table describes the command line options for ServiceTestExecuter.exe:

UI Elements	Description
-test	The full path of the test (required). Specify the test directory—not the solution directory.
-inParams	The full path of an XML file containing the input property values (optional).
-outParams	The full path of an XML file containing the output property values (optional).
-profile	The name of an Test profile (optional). For details, see "How to Define API Test Properties or User/System Variables" on page 141.
-report	The directory in which to store the report.

> **Tip:** To retrieve a list of all of available parameters, run ServiceTestExecuter.exe without any parameters.

The following example runs Test1 using input properties from inParams.xml and output properties from outParams.xml:

```
%ProgramFiles%\HP\Unified Functional Testing\bin> ServiceTestExecuter.exe -test "c:\MyTests
\Test1\" -inParams "c:\MyData\inParams.xml" -outParams "c:\MyData\outParams.xml"
-profile Profile1
```

where the input property file, InParams.xml, has, for example, this structure:

```
<TestParameters>
    <Values>
        <Arguments>
           <a>1</a>
           <b>2</b>
        </Arguments>
    </Values>
</TestParameters>
```

> **Note:** To run a test from the command line, you must save and run the test at least once.

Troubleshooting and Limitations - Run Sessions

Relevant for: GUI and API tests

This section describes troubleshooting and limitations related to running tests.

UFT run sessions

- When running UFT on a remote machine using a Remote Desktop Connection session (RDC) or using Citrix, if the computer on which the application is being tested is logged off or locked, the following problems may occur:

    - The test or component run session may fail.

    - Steps that contain keyboard or focus operations may fail.

    - The Run Results still image capture and/or the Screen Recorder may display a black screen.

    - Steps for which the device level replay is configured to use the mouse (instead of browser events) to run mouse operations may fail. (You set the device level replay using a **Setting.WebPackage("ReplayType")** statement or by setting the **Replay type** option in the Advanced Web Options dialog box.)

    Workaround: If you are using Citrix or a Remote Desktop Connection session to run a test or component, make sure that the computer on which the application is being tested is not logged off or locked.

- When running UFT tests or components on a local machine, if the computer on which the application is being tested is locked, your test run may fail.

    Workaround: Install UFT on a virtual machine (without a screensaver or lock password), and start or schedule your run session on the virtual machine. Then you can lock your local computer without locking the virtual machine.

- It is not recommended to use Test Batch Runner with the UAC (User Account Control) feature set to ON.

Learning objects, running steps, and recording steps (GUI testing only)

- UFT cannot record or run steps if it has limited access to the processes of the application you are testing.

    Workarounds:

    - Make sure that the application you are testing is started by the same Windows user as UFT.

    - Make sure that neither you nor the tested application actively prevent UFT from accessing the application's processes.

- When the title of a window changes during recording, UFT may fail to recognize objects within that window while running the test or component.

  Workaround: Remove the text property from the window's test object description in the Object Repository window.

- UFT cannot record steps in an action opened as read-only.

  Workaround: Make sure that you have read-write access to the action you are recording in.

- **When running steps that contain Insight objects:**

  - The computer session must be active, and the application visible (and not minimized). for the steps to run successfully. This is because Insight uses data from the screen to compare to the images stored with your test.

  - If you are testing an application running on a remote computer, and you use a minimized Remote Desktop Connection window, these steps will fail.

    Workaround: Use a different program (for example, Virtual Network Computing) instead of Remote Desktop Connection to run steps on a remote computer.

  - For Insight object identification to succeed, the display size defined in the operating system and the browser zoom levels (if working on a browser) must be the same when the test runs as they were when the objects were learned or recorded.

  - When using excluded areas in Insight, the included area must contain enough significant content to enable object recognition. If the remaining content is not detailed enough, UFT may locate too few or too many matching controls on the screen.

    Workaround: Do one or both of the following:

    - Use a larger area of the screen for the test object image (make sure to adjust the ClickPoint such that UFT clicks the correct location on the control). For details, see "Change Test Object Image / Add Insight Test Object Dialog Box" on page 1233.

    - Define Visual Relation Identifiers to help identify the control. For details, see "Visual Relation Identifier Dialog Box" on page 1251.

  - When searching for an Insight object within a parent Browser test object, UFT searches within the selected browser tab, not in the whole browser window.

  - **Dual monitor support.** Insight is supported only on the primary monitor. Therefore, if you are working with dual monitors, make sure that your application is visible on the primary monitor when you use UFT for Insight test objects.

- If you selected the **Save movie to results** option in the "Screen Capture Pane (Options Dialog Box > GUI Testing Tab)", the Screen Recorder records a movie of the operations performed on your primary monitor.

Therefore, if you are working with multiple monitors, make sure that your application is fully visible on your primary monitor when recording or running a test or component.

Running Tests (API testing only)

- When you manually add a reference to an external .dll, UFT prompts you to save it locally. To change your preference about a specific referenced file, remove the reference and add it again manually.

- Running tests on remote machines using shared folders, may require adjusting the .NET 2.0 security settings.

  Suggestion: Open the **Control Panel** and locate the **Administrative Tools**, either by browsing or through a search. In the list of **Administrative Tools**, look for the following entry: Microsoft .NET Framework 2.0 Configuration. If it is not present, you must install the .NET Framework 2.0 SDK.

- The Validate Structure checkpoint fails if the expected value is a SOAP Fault and the Web Service call returns an **UnsupportedMediaType** status.

Chapter 23: Running Tests with Virtualized Services

Relevant for: GUI tests and API tests

This chapter includes:

Concepts

Running Tests with Virtualized Services - Overview

Relevant for: GUI tests API tests

Application testing is usually performed on a real deployment of an application. However, sometimes the service upon which an application is based is unavailable or impractical for repeated use or testing. For example, it is impractical to test a flight booking application which requires the entry of a customer's credit card using the real credit card service run in combination with the application, as each time the test is run, the customer's credit card is charged. In such cases, you can replace your application's service with a **virtualized service** during testing.

Using HP Service Virtualization, you create a virtualized service by configuring the behavior of the virtual service to match the expected behavior of the real service. When you are finished creating the service's details in Service Virtualization, the service's details are saved as part of a **virtualization project**. In UFT, you add the virtualization project to a test. The project's settings are saved with the test for future testing sessions.

For details on creating a virtualization project and virtual services, see the *HP Service Virtualization User Guide*.

When designing your test, you insert the virtual service address differently for GUI and API tests:

- For a GUI test, you insert the service address into your application's code in the function where the application calls the real service.

- For an API test, you insert the service's address in place of a URL or service address as a step property.

Before running the test containing the virtualized service, you deploy the service on the Service Virtualization Server. Then, when you run your test, the test runs using the virtual service as needed.

For task details, see "How to Use a Virtualized Service for a UFT Test" on page 667.

For user interface details, see "Virtualized Services Settings Dialog Box" on page 671

This section also includes:

Assigning Data and Performance Models to a Virtualized Service

Relevant for: GUI tests and API tests

When you create and design a virtualized service, part of the configuration process is defining the expected behavior for the service: how quickly it should respond, what the requests and responses to this service should be, and so forth.

Because of this, you define performance models and data models for each virtualized service using Service Virtualization. These models are then saved with your virtualization project.

Later, when you add a virtualization project in a UFT test, the performance models and data models are included with each virtualized service. When you run a test using the virtualized service, you select one of the models to use for a test run.

Performance Models

When you create a virtualization project and add services to the project, you can specify precisely how these services should perform when deployed and run. You can define how quickly the service responds, how often to send requests and responses to the service, the data load for the simulated server, and so forth.

After you create the necessary models in Service Virtualization and add the project to a test in UFT, you can choose which model to use for each test run.

There are a number of different types of performance models for a virtualization project:

User-defined: This model reflects the customized performance settings created in Service Virtualization for a service. Each of the user-defined performance models is available for use in your UFT test.

Offline: This model simulates the unavailability of a service. This model is available for all UFT tests.

None: This model makes the service respond as quickly as possible. This model is available for all UFT tests.

For details on setting and defining performance models for your service, see the *HP Service Virtualization User Guide*

Data Models

In addition to defining performance models for your virtualized service, you can also specify data models. Like performance models, the settings for each model are defined in Service Virtualization, and available for use in UFT tests.

Data models enable you to customize the requests and responses of the service to simulate real service performance. When you create a virtual service, you define the data model, either by providing the requests and responses for the service, or providing a data source that supplies the request and response values. In addition, you can set rules defining the data source use for each of the services and each of the different models.

All data models are user-defined.

For details on setting and defining data models for your service, see the *HP Service Virtualization User Guide*.

Tasks

How to Use a Virtualized Service for a UFT Test

Relevant for: GUI tests and API tests

This task describes how to deploy and use a virtualization project in your test.

This task includes the following steps:

- "Prerequisite - Deploy the Service Virtualization server" below
- "Add a virtualization project to your test in UFT" below
- "Deploy a virtualization project" on the next page
- "Set security credentials for your test to access the virtualization project (optional)" on the next page
- "Set the data and performance models for the virtualization project " on the next page
- "Use the virtualization project in your GUI test" on page 669
- "Use the virtualization project in your API test" on page 669
- "Run the test with a virtualized service" on page 669
- "View the virtualized service details in the run results" on page 669

1. Prerequisite - Deploy the Service Virtualization server

Before you use a virtualization project in a UFT session, you must start the Service Virtualization Server. For details, see the *HP Service Virtualization User Guide*.

2. Add a virtualization project to your test in UFT

a. In UFT, click the **Virtualized Services Settings** button in the toolbar.

b. In the "Virtualized Services Settings Dialog Box" (described on page 671), click the **Add Project** button.

c. In the Add Project dialog box, navigate to your virtualization project or virtualization project archive and click **OK**.

> **Note:** You can open virtualization projects saved in the file system or in an ALM project.

d. In the main Virtualized Services Settings dialog box, click **OK** to add the virtualization project to your UFT test.

> **Note:** You can add multiple virtualization projects to the same test.

> **Tip:** If you change your virtualization project at a later date, update the project in UFT by clicking **Refresh Project** in the main Virtualized Services Settings dialog box.

3. Deploy a virtualization project

After adding the project to the test, and before running the test with a project, you must deploy the project. For subsequent test runs, it is sufficient to check the deployment.

a. In the "Virtualized Services Settings Dialog Box", select the project you want to deploy.

b. Click the **Deploy Project** button. UFT pauses and checks, and deploys (if necessary) the project on the server.

If the project is successfully deployed on the Service Virtualization Server, a ✓ is displayed in the **Deployed Column** of the Virtualized Services Settings dialog box. If the project is not deployed correctly, a ✗ is displayed for the project.

If a project does not deploy correctly, hover over the ✗ to see a description of the problem.

> **Note:** After the initial deployment, you can check the service's deployment before subsequent run sessions by clicking the **Check Deployment** button.

4. Set security credentials for your test to access the virtualization project (optional)

If the Service Virtualization server running the service requires security, you must provide security credentials for UFT to use the deployed service:

a. In the "Virtualized Services Settings Dialog Box", click the **Servers Security** button.

b. In the Servers Security dialog box, enter the user name and password for the servers running your virtualized service.

UFT uses these credentials each time that it accesses the virtualized service during a test run.

5. Set the data and performance models for the virtualization project

For each of your virtualization projects, you can configure how the service uses data when running the virtualized service. Before running a test using the virtualized service, you need to instruct UFT which of the associated data models to use:

a. In the "Virtualized Services Settings Dialog Box", select the virtualization project to use for the current test run.

b. In the **Data Model** and **Performance Model** column, select the name of the data model and performance model from the drop-down list to use for the current test run.

   UFT uses the settings specified in the virtualization project for the service's performance and data usage.

For details on data and performance models in your virtualized service, see "Assigning Data and Performance Models to a Virtualized Service" on page 665.

6. Use the virtualization project in your GUI test

Make sure that your application is configured to use the virtual service address, as specified in the virtualization project. For details on defining virtual service addresses in your virtualization project, see the *HP Service Virtualization User Guide*.

7. Use the virtualization project in your API test

When creating your API test steps, you can use the virtualized services in place of calls or requests to real services, including:

- The URL for your Web Service steps
- The URL for your REST Service steps
- The URL for a HTTP Request or SOAP request step

8. Run the test with a virtualized service

After making all necessary changes for your associated virtualization project, run the test by selecting **Run > Run** or clicking the **Run** button ▷.

When the test runs the step using the virtualized service, it accesses the necessary service as defined in your virtualization project and runs the service.

> **Note:** If you want to disable the virtual services and run the test with the real service, use the **Set all simulated services to standby mode** option in the Virtualized Services Settings dialog box. UFT places all virtualized services in standby and runs the test with a real service.

9. View the virtualized service details in the run results

In the Run Results Viewer, navigate to the virtualization step . The service address is displayed in the result details. as well as details about the service performance (both for the data and performance models).

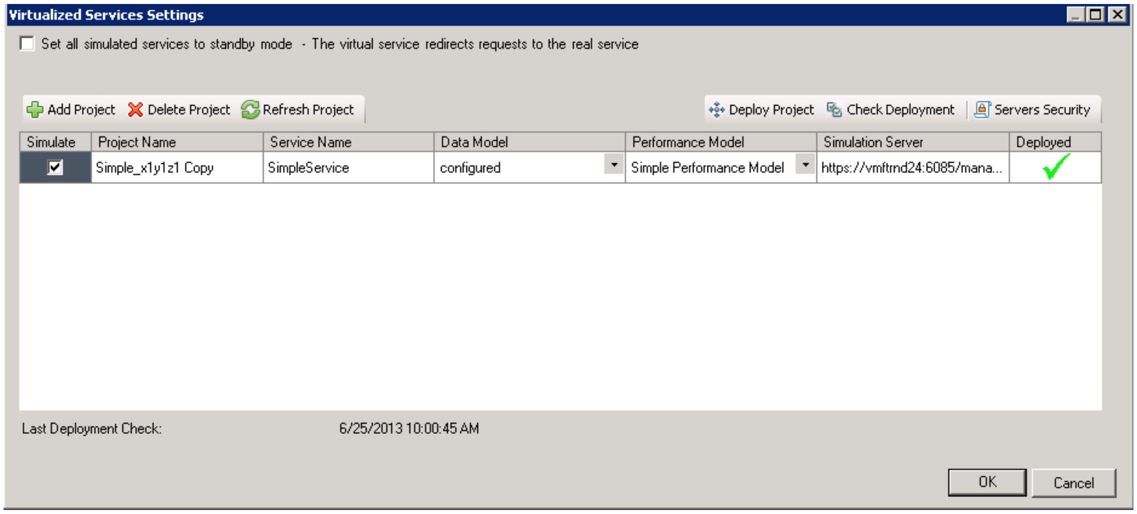
Note: If your service's deployment state was changed after you checked its deployment using the **Check Deployment** button, the run results report the current deployment state in the Virtualization table and the most recent seen deployment state from Virtualized Services Setting dialog box in the service details.

Reference

Virtualized Services Settings Dialog Box

Relevant for: GUI tests and API tests only

This dialog box enables you to assign a virtualization project created in HP Service Virtualization to your test and use this virtualized service during a test run.



☐ Set all simulated services to standby mode - The virtual service redirects requests to the real service

Simulate	Project Name	Service Name	Data Model	Performance Model	Simulation Server	Deployed
☑	Simple_x1y1z1 Copy	SimpleService	configured	Simple Performance Model	https://vmftrnd24:6085/mana...	✓

Last Deployment Check: 6/25/2013 10:00:45 AM

To access	1. Do one of the following: ■ Ensure that a test is in focus in the document pane. ■ In the Solution Explorer, select a test. 2. Click the **Virtualized Services Settings** toolbar button .
Important information	• If you deployed a service with the Service Designer in Service Virtualization, you must stop the service simulation in the Service Virtualization Designer window, before running the test that uses the virtualized service. Failure to do so will result in the service being locked for all other users. • If you loaded a test or a service from the file system or an ALM project and you lost or changed the ALM connection information, any services and projects are reported as missing when you refresh the test. If you check the deployment of the services, they will still work as intended. • You should not call a test that uses the same virtual service as the current test, as the virtualized service is locked by the calling test and the called test cannot run.

Relevant tasks	"How to Use a Virtualized Service for a UFT Test" on page 667
See also	- "Running Tests with Virtualized Services - Overview" on page 665 - "Assigning Data and Performance Models to a Virtualized Service" on page 665

The user elements are described below:

UI Element	Description
Set all simulated services to standby mode	Disables the current deployment of all virtualized services currently associated with your test. When you run your test, the test accesses the real service instead of the virtual service. By default, this option is selected. **Note:** You cannot edit your service details when this option is selected.
Add Project	Opens the Add Project dialog box, which enables you to assign virtualization projects to your test from the file system or from an ALM project. **Note:** You can assign multiple virtualization projects to a test.
Delete Project	Deletes association of the selected projects from the test.
Refresh Project	Updates any changes made in the selected virtualization project. **Caution:** After refreshing a project, you need to check that your server security settings are not lost.
Simulate	Enables you to enable or disable the deployment of the selected virtualized service. When this option is selected, clicking the **Deploy Project** button or the **Check Deployment** is performed for the selected service only.
Project Name	The name of the virtualization project.
Service Name	The name of the virtualized service. **Note:** This name can be different from the name of the virtualization project, as you can have multiple virtual services in the same virtualization project. For details, see the *HP Service Virtualization User Guide*.
Data Model	The data model to use for the current test run. The data models and their properties are defined when designing the virtualization project.
Performance Model	The performance model to use for the current test run. You can choose from a performance model you define in the virtualization project, **Offline**, or **None**.

UI Element	Description
Simulation Server	The address of the server on which the virtualized service is run. **Note:** If the value was configured on the server machine as localhost, be sure to change it in UFT to the actual server name.
Deployed	The status of the selected service. You can see one of the following statuses: - ● **Unknown.** - ✓ **Deployed.** Successfully deployed. - ✗ **Not Deployed.** The service is not deployed on the specified Service Virtualization server. **Tip:** If a project does not deploy, hover over the icon to see a description of the problem.
Deploy Project	Deploys the selected service on the Service Virtualization Server, if it is not already deployed. **Note:** If you are deploying your project on a secure Service Virtualization server and you have not installed the service certificate on your machine, any project that you try to deploy is reported as Not **Deployed**.
Check Deployment	Checks the deployment of the selected project and service.
Servers Security	Opens the Servers Security dialog box, which enables you to enter the user name and password for the Service Virtualization Server running your virtualized service. **Note:** This information is used by UFT to access a secure server when running the virtual service during a test run.

Chapter 24: Debugging Tests and Components

Relevant for: GUI actions, scripted GUI components, function libraries, user code files, and business process tests

This chapter includes:

Concepts

Debugging Overview

Relevant for: GUI actions, scripted GUI components, function libraries, user code files, and business process tests

After you create testing documents such as a test, component, function library, event handler, or user code file, you should check that it runs smoothly, without errors in syntax or logic.

By controlling and debugging your run sessions, you can identify and handle problems in your tests, components, function libraries, registered user functions, event handlers, or user code files.

> In order to debug a function library, you must first associate it with a test or with a component via its application area, and then debug it from that test or component.

UFT provides different options that you can use to detect and isolate defects in adocument. For example:

- You can control the run session using the **Pause** command, breakpoints, and various step commands that enable you to step into, over, and out of a specific step.
- When a run session is suspended, you can use the Debug panes to check and modify the values of code objects and variables and to manually run script or code commands.
- **For GUI testing:**
  - If UFT displays a run error message during a run session, you can click the **Debug** button on the error message to suspend the run and debug the document.
  - If you are working in a test or component, you can use the **Debug from Action** (for tests only) or **Debug from Step** command to begin (or pause) your debug session at a specific point in your test or component. You can also use the **Run to Action** (for tests only) or **Run to Step** command to pause the run at a specific point in your test or component. You can set breakpoints, and then enable and disable them as you debug different parts of your test, component, or function library.

    You can also use the **Run from Action** (for tests only) or **Run from Step** to run your test or component from a selected step or action. This enables you to check a specific section of your application or to confirm that a certain part of your test, component, or function library runs smoothly.
  - To check a specific section of your test, use the **Run Current Action** command. This enables you to check a specific section to ensure that it runs smoothly without running the entire test.

Debugging business process tests consists of adding breakpoints to specific components or flows in your test, and running the test, or pausing the test in the middle of a run session to debug a specific step. For details, see "Breakpoints " on page 681. For details about running business process tests, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

Considerations for Debugging GUI Tests and Components

Relevant for: GUI actions, scripted GUI components, and function libraries

- While a test, component, action, or function library is running in debug mode, it is read-only. You can modify the content after you stop the debug session (not when you pause it). After you implement your changes, you can continue debugging your document.

  > **Note:** If needed, you can enable the function library for editing after you stop the session. Right-click the function library tab in the document pane and select **Enable Editing**. (You cannot enable editing if the function library is locked by another user or checked in to an ALM project.)

- If you perform a file operation (for example, you open a different test or component, or create a new test or component), the debug session stops.

- **When debugging a test:** If a file is called using an **ExecuteFile Statement**, you cannot debug the file or any of the functions contained in the file. In addition, when debugging a test that contains an **ExecuteFile** statement, the execution marker may not be displayed correctly.

  To debug a dynamically loaded function library, use a **LoadFunctionLibrary Statement** statement to load it instead of an **ExecuteFile** statement.

- When you open a test or component, UFT creates a local copy of the external resources that are saved to your ALM project. Therefore, any changes you apply to any external resource that is saved in your ALM project, such as a function library, will not be recognized in the test or component until the test or component is closed and reopened. (An external resource is any resource that can be saved separately from the test or component, such as a function library, a shared object repository, or a recovery scenario.)

  In contrast with this, any changes you apply to external resources saved in the file system, such as function libraries, are implemented immediately, as these files are accessed directly and are not saved as local copies when you open your test.

Considerations for Debugging API User Code Files

Relevant for: User code files

- In order to use theAPI debugging features, you must enable the debugger in the API Testing **General** pane of the Options dialog box (**Tools > Options > API Testing** tab **> General** node). Select the **Run test in debugging mode** check box. For details, see "General Pane (Options

Dialog Box > API Testing Tab)" on page 565.

- While tests are running in debug mode, they are read-only. You can modify user code files after you stop the debug session (not when you pause it).

- When you open a test saved on an ALM project, UFT creates a local copy of the external references that are saved to your ALM project. Therefore, any changes you apply to any external reference that is saved in your ALM project, such as a WSDL file, will not be recognized in the test or component until the test or component is closed and reopened. (An external reference is any resource that can be saved separately from the test or component, such as a WSDL file, XML file, or data table.)

Debug Session Speed

Relevant for: GUI actions and scripted GUI components

During a run session, UFT normally runs steps quickly. While you are debugging a test, component, or function library, you may want UFT to run the steps more slowly so you can pause the run when needed or perform another task.

For task details, see "Slow your debugging session (GUI testing only)" on page 684.

Single Step Commands

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

You can run a single step or step-by-step in a test, component, function library, or user code file by using the **Step Into**, **Step Out**, and **Step Over** commands.

Step Into

Step Into runs only the current step in the active document.

> **Note: (For GUI testing):** When debugging a GUI test, if the current step calls another action or a function, the called action or function is displayed in the document pane. The test or function library pauses at the first line of the called action or function.

Step Out

After using **Step Into** to enter a step or function in a function library or in a user code file, you can use the **Step Out** command. **Step Out** continues the run to the end of the function, or user code file, returns to the calling test, component, or function library, and then pauses the run session at the next line (if one exists).

Step Over

Step Over runs only the current step in the active document.

If the current step calls a user-defined function, the called function is executed in its entirety, but the called function script is not displayed in the document pane. The run session then returns to the calling document and pauses at the next step (if one exists).

> **Note: (For GUI testing):** If the current step calls another action, the called action is displayed in the document pane, and the run session pauses at the first line of the called action (like **Step Into**).

For task details, see "Step into, out of, or over a specific step during a debug session (GUI and API tests only)" on page 684.

Running to a Step and Debugging from a Step

Relevant for: GUI actions and scripted GUI components

You can use the **Run to Action** (tests only), **Run to Step**, and **Debug from Step** commands to instruct UFT to run a test, action, or component (including any associated function library), until it reaches a particular step or action, or to begin debugging from a specific step or action. You can also use the **Debug from Action** (tests only), **Run from Action** (tests only), and **Run from Step** commands to start or continue a run from a particular step or action.

The **Run to Step**, **Debug from Step**, and **Run from Step** commands are available in the **Run** menu, and when you right-click a step in your action or component.

For tests only: The **Run to Action**, **Debug from Action**, and **Run from Action** commands are available when you right-click an action in the test canvas.

Run to Step

You can instruct UFT to run from the beginning of the test, action, or component—or from the current location in the test, action or component—and to stop at a particular step. This is similar to adding a temporary breakpoint to a step. For example, if you want to begin debugging your test, action, or component from a particular step, you may want to run your test, action, or component to that step, as this opens your application to the relevant location.

Debug from Step

You can instruct UFT to begin your debug session from a particular step instead of beginning the run at the start of the test, action, or component. Before you start debugging from a specific step, make sure that the application is open to the location where you want to start debugging. You can begin debugging from a specific step in your test, action, or component when editing a test, action, or component.

For task details, see "Start or pause your debugging session at a specific step or action in your test or component (GUI testing only)" on page 685.

Run from Step

You can instruct UFT to run your test, action, or component from a particular step instead of from the beginning of the test, action, or component. Before you start running from a specific step, make sure that the application is open to the location where you want the run to begin.

For tests:

- In the Editor, the **Run from Step** option runs your test from the selected step until the end of the action (or until it reaches a breakpoint). Using **Run from Step** in this mode ignores any iterations. However, if the action contains nested actions, UFT runs the nested actions for the defined number of iterations of the nested action.
- In the Keyword View, if a single action is displayed, the **Run from Step** option runs your action from the selected step until the end of the action (or until it reaches a breakpoint). If the test flow is displayed, the **Run from Step** option runs the test from the selected step until the end of the test (or until it reaches a breakpoint), as long as all of the actions are internal actions that are local to your test.

  Iterations are handled as follows:

  - If you select an Action node, the **Run from Step** option runs only one iteration of the test, from the selected step until the end of the test. Within the part of the test that runs, UFT runs each action for the number of iterations defined for that action.
  - If you select a step inside an action, the **Run from Step** option runs only one iteration of the action, from the selected step until the end of the action. Within the part of the action that runs, UFT runs any nested actions for the number of iterations defined.

    > **Note:** If your test contains a call to an external action, the run session stops when that action is reached. Similarly, if you use the **Run from Step** option from within an external action, the run stops at the end of that action (or when a breakpoint is reached).

For components: The **Run from Step** option runs your component from the selected step until the end of the component (or until it reaches a breakpoint).

> **Example:**
>
> After you debug a part of your action or component, you may want to skip over a set of steps that you know work correctly, and then continue the debug session from a later step. You can do this by inserting a breakpoint in the step where you want to continue debugging, and then using the **Run from Step** option to run the action or component from the step at which you stopped debugging until it reaches the breakpoint.
>
> Alternatively, you can use the **Run from Step** option to continue the run session from a particular step to the end of the test, action, or component instead of stopping at a breakpoint.

Run to Action (tests only)

You can instruct UFT to run from the beginning of the test until the beginning of the selected action and then pause the run session. For example, if you want to begin debugging your test from a particular action, you may want to run your test until that action, as this opens your application to the relevant location.

Debug from Action (tests only)

You can instruct UFT to begin a debug session, and pause it, at the beginning of the selected action.

Run from Action (tests only)

You can instruct UFT to start a run session from the beginning of the selected action.

Watching the Values of Variables and Properties of Objects During a Run Session

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

You can use the Watch pane and Local Variables pane to view the current value of different code expressions, variables, and object properties in a suspended run session of your test or component. A run session is suspended, for example, if you use the **Run > Pause** command, or when it stops at a breakpoint.

The Local Variables pane displays the current values and types of all variables in the main script of the current action, or in a selected function in your test, function library, or user code files.

The Watch pane displays the current values and the types of code expressions and objects that you add to the pane.

As you continue stepping through the subsequent steps in your test, function library, or user code file, UFT automatically updates the Watch pane and Local Variables pane with the current value for any variable or expression whose value changes. In addition, UFT reevaluates the information displayed in the Watch pane and Local Variables pane as you make changes in the context of your debug session, as selected in the Call Stack pane,

You can add any of the following types of expressions to the Watch pane:

- The name of a GUI test object
- The name of a variable
- The name of a property
- Any other type of code expression

> **Caution:** UFT runs the expressions in the Watch pane to evaluate them. Therefore, do not add a method or any expression whose evaluation could affect the state of the test or any GUI test object, as this can lead to unexpected behavior of your test, component, function library, or user code file.

Expressions added to the Watch pane are saved with the document and updated accordingly as you make changes to your document.

For task details, see "Check and modify the values of variables and code expressions during a debug session" on page 685.

Breakpoints

Relevant for: GUI actions, scripted GUI components, function libraries, user code files, and business process tests

You can use breakpoints to instruct UFT to pause a run session at a predetermined place in adocument. UFT pauses the run when it reaches the breakpoint, before executing the step. You can then examine the effects of the run up to the breakpoint, make any necessary changes, and continue running the document from the breakpoint.

Breakpoints are saved with your document and are maintained even after you close and reopen UFT. UFT maintains the breakpoints inserted in a test or component per user on a given computer.

You can use breakpoints to:

- Suspend a run session and inspect the state of your application
- Mark a point from which to begin stepping through adocument using the "Single Step Commands" on page 677

For task details, see "How to Use Breakpoints " on page 687.

Note:

- If you a run a test using the Run automation method, the test does not stop at breakpoints even if they are saved in the test.
- If you run a test with breakpoints using the Run automation method, the breakpoints remain visible but are ignored during the test run.
- If you are running a test from the ALM Test Lab module in **hidden mode** (as specified in the UFT Remote Agent, UFT will not stop the test at the breakpoints.
- If you are running a test from the ALM Test Plan module not in hidden mode, the test stops at breakpoints if you select the **Run Test Sets in debug mode** option in the UFT Remote Agent

For details on Remote Agent settings, see "Remote Agent Settings Dialog Box" on page 745.

Enabling and Disabling Breakpoints

Relevant for: GUI actions, scripted GUI components, function libraries, user code files, and business process tests

You can instruct UFT to ignore an existing breakpoint during a debug session by temporarily disabling the breakpoint. Then, when you run your document, UFT runs the step containing the breakpoint, instead of stopping at it. When you enable the breakpoint again, UFT pauses there during the next run. This is particularly useful if your document contains many steps or events, and you want to debug a specific part of it.

You can enable or disable breakpoints individually or all at once. For example, suppose you add breakpoints to various steps throughout your document, but for now, you want to debug only a specific part of it or a specific event. You could disable all breakpoints in your document and then enable breakpoints only for specific steps or in specific event handlers. After you finish debugging that section, you could disable the enabled breakpoints, and then enable the next set of breakpoints (in the section or event you want to debug). Because the breakpoints are disabled and not removed, you can find and enable any breakpoint, as needed.

Enabled breakpoint. An enabled breakpoint is indicated by a filled red circle icon (●) in the left margin adjacent to the selected step.

Disabled breakpoint. A disabled breakpoint is indicated by an empty circle icon (○) in the left margin adjacent to the selected step.

When UFT runs a test without stopping on breakpoints, such as when running in hidden mode, the Editor indicates that the breakpoint is not in use.

Run Errors

Relevant for: GUI actions, scripted GUI components, and function libraries and user code files

Run errors are treated differently for GUI and API tests.

For GUI testing: There are two types of Run Error message boxes that can be displayed during a run session:

- A pure syntax error. When a syntax run error message box is displayed, click **OK** in the message box and address the error.

- The other message box can be displayed in a number of situations. It offers information about the error and a number of buttons for dealing with errors encountered.

For user interface details, see "Run Error Message Box" on page 701.

For API testing: Run-time errors are reported in the run results.

Tasks

How to Debug Your Test, Component, Function Library, or User Code File

Relevant for: GUI actions, scripted GUI components, function libraries, user code files, and business process tests

This task describes different ways you can control and debug your run sessions so you can identify and handle problems in your documents.

To practice this task, see "How to Debug a GUI Action or a Function - Exercise" on page 689 (for GUI testing) or"How to Debug an API User Code File - Exercise" on page 693 (for API testing).

> **Note: (for GUI testing):** You can use the Screen Recorder to capture a movie of your application as it is being tested. For details, see the section on the Run Results Viewer Screen Recorder Pane (described in the *HP Run Results Viewer User Guide*).

This task contains the following sections:

- "Prerequisites" on the next page
- "Slow your debugging session (GUI testing only)" on the next page
- "Step into, out of, or over a specific step during a debug session (GUI and API tests only)" on the next page
- "Start or pause your debugging session at a specific point (GUI testing and business process testing only)" on the next page
- "Start or pause your debugging session at a specific step or action in your test or component (GUI testing only)" on page 685
- "Use breakpoints in your document" on page 685
- "Handle run errors (GUI testing only)" on page 685
- "Check and modify the values of variables and code expressions during a debug session" on page 685
- "Manually run code commands during a debug session" on page 686
- "View the current call stacks" on page 687
- "View currently running threads (API testing only)" on page 687
- "View the loaded modules associated with the run session (API testing only)" on page 687

Prerequisites

You must have the Microsoft Script Debugger installed to run tests or components in debug mode. If it is not installed, you can use the UFT Additional Installation Requirements Utility to install it. (Select **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Additional Installation Requirements**.)

For GUI testing: To debug components in UFT, you must enable integration between UFT and your ALM project. In UFT, select **Tools > Options > GUI Testing** tab **> Run Sessions** node and select the **Allow other HP products to run tests and components** check box.

For API testing: To debug API tests, you must enable the debugger. Select **Tools > Options > API Testing** tab **> General** node and select **Run test in debugging mode**.

> **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

Slow your debugging session (GUI testing only)

In the **Test Runs** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Test Runs** node), specify the time (in milliseconds) UFT pauses between each step by modifying the **Delay each step execution by** option. For details on the Run pane options, see "Test Runs Pane (Options Dialog Box > GUI Testing Tab)" on page 539.

Step into, out of, or over a specific step during a debug session (GUI and API tests only)

- To use the **Step Into** command select **Run > Step Into**, click the **Step Into** button , or press **F11.**
- To use the **Step Out** command select **Run > Step Out**, click the **Step Out** button , or press **SHIFT+F11**.
- To use the **Step Over** command select **Run > Step Over**, click the **Step Over** button , or press **F10**.

For details, see "Single Step Commands" on page 677.

For GUI testing: For an example of using single step commands, see "How to Step Into, Out of, or Over a Specific Step in a GUI Test - Exercise" on page 697.

Start or pause your debugging session at a specific point (GUI testing and business process testing only)

- To temporarily suspend a run session, click the **Pause** button ⏸ or select **Run > Pause**. A paused document stops running when all previously interpreted steps have been run.
- To resume running a paused run, click the **Run** ▶ button or select **Run > Run**. The run continues from the point it was suspended.

Start or pause your debugging session at a specific step or action in your test or component (GUI testing only)

- To instruct UFT to run to a particular step, select the step in your document at which you want UFT to stop and select **Run > Run to Step** or press **CTRL+F10**.
- To instruct UFT to run from a particular step, select the step at which you want UFT to start the run and select **Run > Debug from Step**.

> **Note:** These commands can also be used to stop at a specific action. Right-click an action in the canvas and select **Run to Action**, **Debug from Step**, or **Run from Action**.

Use breakpoints in your document

For details, see "How to Use Breakpoints " on page 687.

> **Note:**
>
> - If you a run a test using the Run automation method, the test does not stop at breakpoints even if they are saved in the test.
> - If you run a test with breakpoints using the Run automation method, the breakpoints remain visible but are ignored during the test run.
> - If you are running a test from the ALM Test Lab module in **hidden mode** (as specified in the UFT Remote Agent, UFT will not stop the test at the breakpoints.
> - If you are running a test from the ALM Test Plan module not in hidden mode, the test stops at breakpoints if you select the **Run Test Sets in debug mode** option in the UFT Remote Agent
>
> For details on Remote Agent settings, see "Remote Agent Settings Dialog Box" on page 745.

Handle run errors (GUI testing only)

For details, see "Run Error Message Box" on page 701.

Check and modify the values of variables and code expressions during a debug session

- To add an expression to the Watch Pane, do one of the following:
  - Click the **Add New Watch Expression** button ➕ in the "Watch Pane" (described on page 297) and enter the name of the expression in the Add New Watch dialog box.
  - **For GUI actions, scripted GUI components, and function libraries only:** Highlight the selected expression and select **Run > Add to Watch** right-click the expression and select **Add to Watch** from the context menu.

- To remove an expression from the "Watch Pane", select the row in the Watch pane that you want to remove and press the **DELETE** key on your keyboard or press the **Delete** button ✖.
- To view the current values for all variables up to the current step in the document, use the "Local Variables Pane".

You can also change the value of a variable or property manually by running code commands in the Console pane.

For more details, see:

- "Watch Pane" on page 297
- "Local Variables Pane" on page 288
- "Console Pane" on page 292

Note: (**GUI testing only**)

- To add an **identification property** to the Watch pane, you must use an expression that calls GetROProperty. This enables you to watch the run-time value of the object's identification property. For example, to watch the value currently displayed in the Calculator application, you can add the expression:

  ```
  Window("Calculator").WinEdit("Edit").GetROPRoperty("text")
  ```

- You cannot modify the run-time value of an object's identification property from the Watch pane or the Local Variables pane.
- You can add an expression to the Watch pane from the Editor or from a function library, but not from the Keyword View.
- To expand the ability of UFT to display information on test objects, it is recommended to register PDM from Internet Explorer (for versions of Internet Explorer 8 or higher and Visual Studio 2008).Enter the following in the command line to register the .dll:

  ```
  regsvr32 "%ProgramFiles%\Internet Explorer\pdm.dll"
  ```

Manually run code commands during a debug session

For details, see "Console Pane" on page 292.

View the current call stacks

To view the currently running call stacks in your run session, select **View > Debug > Call Stack**. You can double-click on a stack name in the pane to navigate directly to the line of code that begins the call stack.

> **Note:** If the location of the call stack is not in a currently open file, UFT opens the relevant file.

View currently running threads (API testing only)

To view the code threads currently running in the run session, select **View > Debug > Threads**. You can double-click on a thread's name in the "Threads Pane (API Testing) " (described on page 287) to navigate directly to the beginning of the thread.

> **Note:** If the file containing the beginning of the selected thread is not open, then UFT opens the relevant file.

View the loaded modules associated with the run session (API testing only)

To view the associated loaded modules for your run session, select **View > Debug > Loaded Modules.**

How to Use Breakpoints

Relevant for: GUI tests, scripted GUI components, function libraries, user code files, and business process tests

The following steps describe how to set breakpoints, and temporarily enable or disable them. After you finish using them, you can remove them from your document.

This task includes the following sections:

- "Set a breakpoint" below
- "Enable or disable a breakpoint" on the next page
- "Enable or disable all breakpoints" on the next page
- "Remove a single breakpoint or all breakpoints" on the next page

Set a breakpoint

To set a breakpoint, do one of the following:

- Click in the left margin of a step in the document where you want the run to stop.
- Select the line where you want the run to stop and select **Run > Insert/Remove Breakpoint**.

The breakpoint symbol ● is displayed in the left margin adjacent to the selected step.

Note:

- If you a run a test using the Run automation method, the test does not stop at breakpoints even if they are saved in the test.
- If you run a test with breakpoints using the Run automation method, the breakpoints remain visible but are ignored during the test run.
- If you are running a test from the ALM Test Lab module in **hidden mode** (as specified in the UFT Remote Agent, UFT will not stop the test at the breakpoints.
- If you are running a test from the ALM Test Plan module not in hidden mode, the test stops at breakpoints if you select the **Run Test Sets in debug mode** option in the UFT Remote Agent

For details on Remote Agent settings, see "Remote Agent Settings Dialog Box" on page 745.

Enable or disable a breakpoint

To enable/disable a specific breakpoint, do one of the following:

- Right-click the step containing the breakpoint and select **Enable/Disable Breakpoint**.
- In the " Breakpoints Pane", select the breakpoint you want to enable or disable and press the **Disable/enable breakpoint** button .

Enable or disable all breakpoints

To enable/disable all breakpoints, select **Run > Enable/Disable All Breakpoints.** If at least one breakpoint is enabled, UFT disables all breakpoints in the document. Alternatively, if all breakpoints are disabled, UFT enables them.

Remove a single breakpoint or all breakpoints

To remove a single breakpoint, click the breakpoint icon in the left margin of the step. The breakpoint symbol is removed from the left margin of the document.

To remove all breakpoints, do one of the following:

- Select **Run > Clear All Breakpoints.**
- In the " Breakpoints Pane", click the **Remove all** button .
- In the " Breakpoints Pane", right-click and select **Remove all**.

All breakpoint symbols are removed from the left margin of the document.

Navigate to a specific breakpoint

1. In the " Breakpoints Pane", select the specific breakpoint to which you want to navigate.

2. Do one of the following:

   - Double-click the line containing the breakpoint name.
   - Click the **Go to source** button .
   - Right-click the line containing the breakpoint and select **Go to Source**.

   The cursor flashes in the main document window in the line containing the breakpoint.

How to Debug a GUI Action or a Function - Exercise

Relevant for: GUI actions. scripted GUI components, and function libraries

In this exercise, you create and debug an action or a function, to practice using some of UFT's debugging capabilities for GUI tests.

Suppose you create an action or a function that defines variables that are used in other parts of your test or function library. You can add breakpoints to the action or function to see how the value of the variables changes as the test or function library runs. To see how the test or function library handles the new value, you can also change the value of one of the variables during a breakpoint.

> **Note:** For a task related to this exercise, see "How to Debug Your Test, Component, Function Library, or User Code File" on page 683.

This exercise includes the following steps:

- "Create a new action or function" below
- "Associate the function library with a test or application area (function libraries only)" on the next page
- "Add a call to the function in your test or component (function libraries only)" on page 691
- "Add breakpoints" on page 691
- "Begin running the test or component" on page 691
- "Check the value of the variables in the debug panes" on page 691
- "Check the value of the variables at the next breakpoint" on page 692
- "Modify the value of a variable using the Console pane" on page 692
- "Repeat a command from the command history" on page 692

1. **Create a new action or function**

   a. Do one of the following:

- **For tests:** Create or open a test. In addition, as an optional step, you can create or open a function library.
- **For components:** Create or open a function library. For details on creating a function library, see "How to Create and Manage Function Libraries" on page 1041.

b. Create a new function called **SetVariables**. For details on working with functions, see "User-Defined Functions and Function Libraries" on page 1029.

Enter the VBScript code in the Editor of your action or the function library, as follows:

Action

```
Dim a
a="hello"
b="me"
MsgBox a
```

Function Library

```
Function SetVariables

Dim a
a="hello"
b="me"
MsgBox a

EndFunction
```

For more details on the Editor, see "Programming in GUI Testing Documents in the Editor" on page 994.

> **Note:** If you are working only with an action in the Editor, skip to Add Breakpoints. If you are working in a function library (in a test or component), proceed to the next step.

2. Associate the function library with a test or application area (function libraries only)

a. If you are working with a component, make sure the application area associated with your component is open.

b. Bring the function library into focus.

c. Right-click the function library document tab and select **Associate Library '<Function Library Name>' with '<Test/ Application Area Name>'**. UFT associates the function library with your test or application area.

> **Note:** For additional details on how to associate a function library, see "How to Manage Function Library Associations" on page 1045.

3. ### Add a call to the function in your test or component (function libraries only)

   Do one of the following:

   - **For tests:** Add a call to the function by typing the following in the Editor:

     ```
     SetVariables
     ```

   - **For components:** Add a call to the function by inserting a new operation and choosing **SetVariables** from the **Operation** list.

4. ### Add breakpoints

   Add breakpoints at the lines containing the text b="me" and MsgBox a. For details on adding breakpoints, see "Breakpoints " on page 681.

5. ### Begin running the test or component

   Run the test or component. The test or component stops at the first breakpoint, before executing that step (line of script).

6. ### Check the value of the variables in the debug panes

   a. Select **View > Debug > Watch** to open the Watch pane.

   b. In the Editor of your test action or in the function library, highlight the variable **a** and select **Run > Add to Watch**.

      UFT adds the variable **a** to the Watch pane. The **Value** column indicates that the value of **a** is currently **"hello"**, because the breakpoint stopped after the value of variable **a** was initiated. The **Type** column indicates that **a** is a **String** variable.

   c. In the Editor displaying your test action or in the function library, highlight the variable **b** and select **Run > Add to Watch**.

      UFT adds the variable **b** to the Debug Watch pane. The **Value** column indicates **<Variable is undefined: 'b'>** (and the **Type** column displays **Incorrect expression**), because the test or component stopped before variable **b** was declared.

   d. Select **View > Debug > Local Variables** to open the Local Variables pane.

      For tests: Only variable **a** is displayed (with the value **"hello"**), because **a** is the only variable that was initiated up to this point.

For components: Both **SetVariables** (with the value **Empty**) and variable **a** (with the value **"hello"**) are displayed.

For both tests and components, variable **b** is not displayed because the test or component stopped before variable **b** was declared.

7. **Check the value of the variables at the next breakpoint**

   Click the **Run** button to continue running the test or component.

   The test or component stops at the next breakpoint. Note that the values of variables **a** and **b** were both updated in the Watch and Local Variables panes.

8. **Modify the value of a variable using the Console pane**

   a. Select **View > Debug > Console**.

   b. At the command prompt at the bottom of the pane, enter:
   if b="me" then a="b is me" else a="b is you" end if
   Then press ENTER on the keyboard.

   c. Click the **Local Variables** pane to verify that the value of variable **a** was updated according to the command you entered and now displays the value: "b is me"

   d. Click the **Run** button to continue running the test or component.

   The message box that opens displays "b is me" (which is the modified value of **a**). This indicates that you successfully modified the values of both **a** and **b** using the Debug Console pane.

   e. Click **OK** to close the message box.

9. **Repeat a command from the command history**

   a. Remove the first breakpoint and run the test or component again.

   When the test or component stops at the breakpoint (before displaying the message box), modify the value of variable **b** in the Console pane by running the command b="not me".

   b. In the Console pane, highlight the command line that reads if b="me" then a="b is me" else a="b is you". Then right-click and select **Copy**.

   c. In the command prompt, right-click and select **Paste**.

   d. Press ENTER to run the command, and then click the **Run** button to complete the test or component run.

   A message box saying "b is you" opens. Click **OK** to close the message box.

How to Debug an API User Code File - Exercise

Relevant for: User code files

In this exercise, you create and debug an API user code file to practice using some of UFT's debugging capabilities for API tests.

> **Note:** For a task related to this scenario, see "How to Debug Your Test, Component, Function Library, or User Code File" on page 683.

This scenario includes the following steps:

- "Create test steps" below
- "Set properties for the math steps" below
- "Create parameters for the Custom Code activity" on the next page
- "Link the Custom Code activity to existing steps" on the next page
- "Create events for the Custom Code activity" on page 695
- "Run the test" on page 695
- "Check the value of the variables at the first breakpoint" on page 695
- "Add a variable to the Watch Pane" on page 696
- "Check the value of the variables at the next breakpoint" on page 696

1. Create test steps

a. Create an API test.

b. From the "Toolbox Pane User Interface (API Testing) ", drag the **Add** activity, **Multiply** activity, and the **Custom Code** activity to the canvas.

2. Set properties for the math steps

a. In the **Input/Checkpoints** tab , in the Input pane, enter the values for the **Add4** operation.

   - In the **Value** column for **A**, enter 10.
   - In the **Value** column for **B**, enter 6.

b. In the **Input/Checkpoints** tab, enter the values for the **Multiply** operation.

- In the **Value** column for **A**, enter 2.
- In the **Value** column for **B**, enter 4.

3. Create parameters for the Custom Code activity

a. In the "Add Input/Output Property/Parameter Dialog Box (API Testing)" (described on page 1775), enter the details for the input parameter.

- In the **Name** field, enter AddResult.
- In the **Type** field, select String from the drop-down list (if it is not already selected).

A new input property called **AddResult** appears in the **Input** pane inside the **Input/Checkpoints** tab.

b. Create another input parameter called **MulResult**. This time, in the "Add Input/Output Property/Parameter Dialog Box (API Testing)":

- In the **Name** field, enter MulResult.
- In the **Type** field, select String from the drop-down list (if it is not already selected).

A new input property called **MulResult** appears in the **Input** pane inside the **Input/Checkpoints** tab.

c. Click the **Add** button again and select **Add Output Property**.

d. In the "Add Input/Output Property/Parameter Dialog Box (API Testing)" (described on page 1775), enter the details for the output parameter.

- In the **Name** field, enter Result.
- In the **Type** field select Decimal from the drop-down list.

e. In the **Checkpoints** pane, enter the value of 128.

4. Link the Custom Code activity to existing steps

a. In the "Select Link Source Dialog Box (API Testing)" (described on page 1840), select the **AddActivity** step. In the right pane, select **Result** and click **OK.**

The link source Step.OutputProperties.AddActivity<number>.Result appears in the **AddResult** row.

b. In the dialog, select the **Multiply** step. In the right pane, select **Result** and click **OK**.

   The link source Step.OutputProperties.MultiplyActivity<number>.Result appears in the **MulResult**.

5. **Create events for the Custom Code activity**

   a. In the Properties pane, select the **CustomCode** activity from the drop-down list or by clicking the **CustomCode** activity in the canvas.

   b. In the Properties pane, select the **Events** tab.

   c. In the "Events Tab (Properties Pane - API Testing)" (described on page 409), create a default handler for **ExecuteEvent** and **AfterExecuteStepEvent**. Two events, CodeActivity<number>_OnExecuteEvent and CodeActivity<number>_OnAfterExecuteEvent are added to the TestUserCode.cs file.

      For details on API testing event handlers, see "Writing Code for API Test Events - Overview" on page 1938.

   d. In the TestUserCode.cs file, enter the following text in the CodeActivity_OnExecuteEvent:

```
decimal AddResult = AddActivity.Result;
decimal MulResult = MultiplyActivity.Result;
CodeActivity<number>.Output.Result = AddResult*MulResult;
```

   e. Enter a breakpoint on the last line of this method.

   f. In the TestUserCode.cs file, enter the following text in the CodeActivity<number>_OnAfterExecuteStepEvent:

```
decimal result = CodeActivity<number>.Output.Result;
```

   g. Enter a breakpoint on the last line of this method.

   h. Save the test.

6. **Run the test**

   Select **Run > Run** or press F5.

7. **Check the value of the variables at the first breakpoint**

   a. When the test run stops at the first breakpoint, select **View > Debug > Local Variables** to open the "Local Variables Pane".

   b. In the Local Variables pane, see the current values of the **Add** and **Multiply** activity. The current values should be 16 for the **AddActivity** and 8 for the **MultiplyActivity**.

You can expand the notes on various rows to see the variable values of the different items used in your test run.

8. Add a variable to the Watch Pane

a. In the **TestUsercode.cs** tab, highlight the text AddResult.

b. Open the Watch pane by selecting **View > Debug > Watch**.

c. In the "Watch Pane" (described on page 297), click the **Add New Watch Expression** button ➕ and enter Add Result.

A line with the expression AddResult, with a value of 16, and type Decimal appears.

d. Click the **Add New Watch Expression** button again to add the variable **MulResult** to the Watch pane. The pane should display the expression MulResult, with a value of 8, and type Decimal.

9. Check the value of the variables at the next breakpoint

a. Continue the run session by selecting **Run > Continue** or pressing F5.

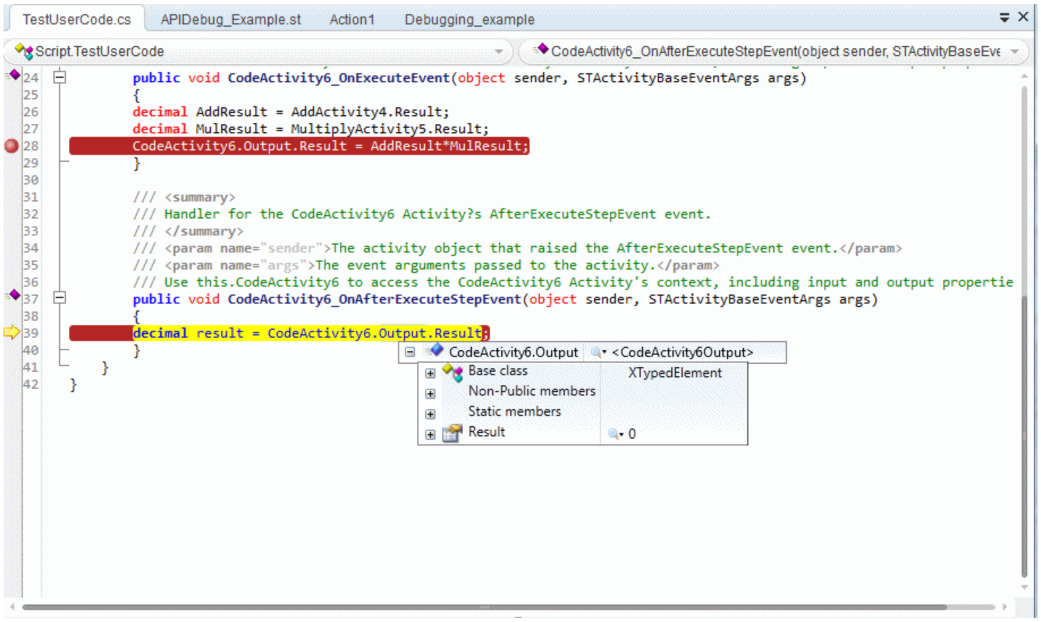
b. When the run session pauses at the next breakpoint, highlight the text CodeActivity<number>.Output.Result and add it to the Watch pane.

The pane displays that the value of this variable is 128.

Note that the **AddResult** and **MulResult** values, which you added in the previous step to the Watch pane, are undefined with a type Incorrect Expression. This is because these values are present and relevant to the current event.

c. Click the Local Variables tab. Note that the line Result displays a value of 128, since the custom code entered earlier noted that Result is equal to CodeActivity6.Output.Result, which was equal to AddResult*MulResult.

In addition, if you hover over the variable names in the Editor in the paused run session, an expandable tooltip displaying the current value of the variable and its properties can be viewed.



```
        public void CodeActivity6_OnExecuteEvent(object sender, STActivityBaseEventArgs args)
        {
        decimal AddResult = AddActivity4.Result;
        decimal MulResult = MultiplyActivity5.Result;
        CodeActivity6.Output.Result = AddResult*MulResult;
        }

        /// <summary>
        /// Handler for the CodeActivity6 Activity?s AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEvent event.</param>
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.CodeActivity6 to access the CodeActivity6 Activity's context, including input and output propertie
        public void CodeActivity6_OnAfterExecuteStepEvent(object sender, STActivityBaseEventArgs args)
        {
        decimal result = CodeActivity6.Output.Result;
        }
    }
}
```

d. Select **Run > Continue** or click F5 to complete the run session and view the run results.

How to Step Into, Out of, or Over a Specific Step in a GUI Test - Exercise

Relevant for: GUI actions, scripted GUI components, and function libraries

In this exercise, you create a sample function library and run it (from a test or component) using the **Step Into**, **Step Out**, and **Step Over** commands.

> **Note:** For a task related to this exercise, see "How to Debug Your Test, Component, Function Library, or User Code File" on page 683.

This exercise includes the following steps:

- "Create the sample function library and test (tests only)" on the next page
- "Create the sample function library and component (components only)" on the next page
- "Run the function library from your test or component, and use the Step Into, Step Out, and Step Over commands" on page 699

1. **Create the sample function library and test (tests only)**

   > **Note:** This procedure is relevant for test actions and function libraries only. If you are working with a component, see the relevant "Create the sample function library and component (components only)" belowfor components below.

   a. Select **File > New > Function Library** to open a new function library.

   b. In the function library, enter the following lines exactly:

   ```
   public Function myfunc()
   msgbox "one"
   msgbox "two"
   msgbox "three"
   ```

   The End Function is automatically added by UFT.

   c. Save the function library to your ALM project or to the file system, with the name SampleFL.qfl. (For details, see "How to Create and Manage Function Libraries" on page 1041.)

   d. Select **File > New > Test** and select **GUI Test** to open a new test.

   e. Click the document tab for the SampleFL.qfl function library to bring it into focus.

   f. Right-click the function library document tab and select **Associate Library 'SampleFL.qfl' with 'Test'** to associate the function library with your test.

   g. Click the document tab for the action you created to bring it into focus. Click the Editor/Keyword View toggle button to display the Editor and enter the following lines exactly:

   ```
   myfunc
   myfunc
   myfunc
   endOfTest="true"
   ```

2. **Create the sample function library and component (components only)**

   > **Note:** This procedure is relevant for components and function libraries only. If you are working with a test, see the relevant "Create the sample function library and test (tests only)" abovefor tests, above.

a. Select **File > New > Function Library** to open a new function library.

b. In the function library, enter the following lines exactly:

```
public Function myfunc()
msgbox "one"
msgbox "two"
msgbox "three"
```

The End Function is automatically added by UFT.

c. Save the function library to your ALM project or to the file system, with the name SampleFL.qfl. (For details, see "How to Create and Manage Function Libraries" on page 1041.)

d. Select **File > New > Application Area** to open a new application area.

e. Save the application area in your ALM project with the name **SampleAA**. (For details, see "How to Create and Manage Application Areas" on page 2097.)

f. Click the document tab for the SampleFL.qfl function library to bring it into focus.

g. Right-click on the function library document tab and select **Associate Library 'SampleFL.qfl' with 'SampleAA'** to associate the function library with the open application area.

h. Select **File > New > Business Component** and select **GUI Keyword Component**. In the "New <Document> Dialog Box" (described on page 152), associate the component with the **SampleAA** application area that you created and saved in your ALM project.

i. In the component, insert four identical steps. For each step:

   - In the **Item** cell, select **Operation**.

   - In the **Operation** cell, select **myfunc**.

3. Run the function library from your test or component, and use the Step Into, Step Out, and Step Over commands

a. Select the first step of the action or component (the first call to the myfunc function) and add a breakpoint by pressing **F9** (**Insert/Remove Breakpoint**). The breakpoint symbol is displayed in the left margin ● . For details, see "Breakpoints " on page 681.

b. Run the test or component. The test or component pauses at the breakpoint.

c. Press **F11** (**Step Into**). The execution arrow points to the first line (msgbox "one") of the function in the function library.

d. Press **F11** (**Step Into**) again. A message box displays the text one.

e. Click **OK** to close the message box. The execution arrow moves to the next line in the function.

f. Continue pressing **F11** (**Step Into**) (and pressing **OK** on the message boxes that open) until the execution arrow leaves the function and is pointing to the second step in the action or component (the second call to the myfunc function).

g. Press **F11** (**Step Into**) to enter the function again. The execution arrow points to the first msgbox line within the function.

h. Press **SHIFT+F11** (**Step Out**). Close each of the message boxes that opens. Notice that the execution arrow continues to point to the first line in the function until you close the last of the three message boxes. After you close the third message box, the execution arrow points to the next line in the action or component (the third call to the myfunc function).

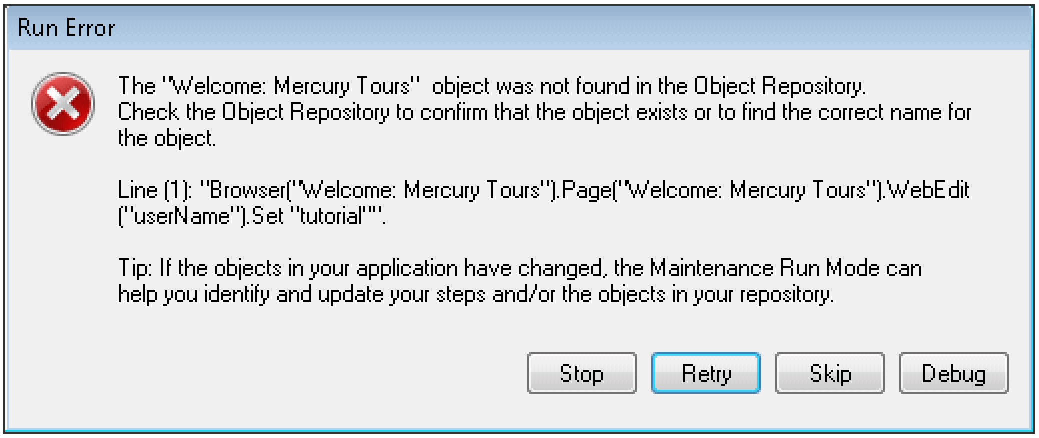
i. Press **F10** (**Step Over**). The three message boxes open again—this time, in the Keyword View. The execution arrow remains on the same step in the action or component until you close the last of the three message boxes. After you close the third message box, the execution arrow points to the next step in the action or component.

Reference

Run Error Message Box

Relevant for: GUI actions, scripted GUI components, and function libraries

This message box enables you to handle run errors during a run session.



Run Error

The "Welcome: Mercury Tours" object was not found in the Object Repository.
Check the Object Repository to confirm that the object exists or to find the correct name for the object.

Line (1): "Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "tutorial"".

Tip: If the objects in your application have changed, the Maintenance Run Mode can help you identify and update your steps and/or the objects in your repository.

To access	This message box is displayed when there is a run error during a run session.
Relevant tasks	"How to Debug Your Test, Component, Function Library, or User Code File" on page 683
See also	"How to Debug a GUI Action or a Function - Exercise" on page 689

User interface elements are described below:

UI Elements	Description
Stop	Stops the run session. The run results are displayed if UFT is configured to show run results after the run.
Retry	UFT attempts to perform the step again. If the step succeeds, the run continues.
Skip	UFT skips the step that caused the error, and continues the run from the next step.
Debug	UFT suspends the run, enabling you to debug the relevant test, component, function library, or event handler. You can perform any of the debugging operations described in this chapter. After debugging, you can continue the run session from the step where the action component, or function library stopped, or you can use the step commands to control the remainder of the run session.
Help	Opens the UFT troubleshooting Help for the displayed error message. After you review the Help topic, you can select another button in the error message box.

Run/Debug Session Toolbar

Relevant for: GUI actions, scripted GUI components, function libraries, and user code files

This toolbar is displayed when you begin a run or debug session.

To access	Start a run or debug session.
Important information	The toolbar buttons described are unique to the Run/Debug Session toolbar. Most of the toolbar buttons displayed during a run or debug session are common to the toolbar displayed when designing or editing a test or component.
Relevant tasks	"How to Debug Your Test, Component, Function Library, or User Code File" on page 683
See also	• "How to Debug a GUI Action or a Function - Exercise" on page 689 • "How to Debug an API User Code File - Exercise" on page 693

User interface elements are described below:

Toolbar button	Name	Description
▷	**Run**	Starts a run session or continues a paused run or debug session.
■	**Stop**	Stops the current run session.
⏸	**Pause**	Pauses the current run session.
	Step Into	Enables you to run only the current step in your document. For details, see "Step Into" on page 677.
	Step Over	Enables you to run only the current step in the document. For details, see "Step Over" on page 677.
	Step Out	Continues the current run session after the current step until the end of the current function. For details, see "Step Out" on page 677.
	Solution Explorer Pane	Displays the Solution Explorer pane as the active pane.
	Toolbox Pane	Displays the Toolbox pane as the active pane.
	Data Pane	Displays the Data pane as the active pane.

Toolbar button	Name	Description
	Properties Pane	Displays the Properties pane as the active pane.
	Debug Pane	Opens a drop-down list of the debug panes you can bring into focus. For details, see "Debug Pane Interface" on page 273.
	Output Pane	Displays the Output Pane as the active pane.
	Errors Pane	Displays the Errors pane as the active pane.
	Tasks Pane	Displays the Tasks pane as the active pane.
	Active Screen	Displays the Active Screen pane as the active pane.

Troubleshooting and Limitations - Debugging

Relevant for: GUI actions and scripted GUI components, and function libraries

This section describes troubleshooting and limitations for debugging.

- If a run-time error occurs during a run session, you can click **Skip** in the error message box that opens, to skip the problematic step and continue the run session. However, during a debugging session, if you set a breakpoint on a line immediately after a line that causes a run-time error, UFT does not stop at this breakpoint when the run session continues after the error.

  Workaround: Set the breakpoint at least two lines after the line that causes the run-time error.

- When debugging a test that is running from ALM, use the F9 key and **Debug** menu command to toggle breakpoints. Clicking the left banner in the Editor to toggle breakpoints is not supported in this scenario.

- **GUI testing only:** UFT may stop responding during a debug run session in the following scenario: If there is a breakpoint at a function call from an associated function library, and you open the function library after the run stops at the breakpoint, and then you insert or remove a breakpoint on the first line of the called function.

- **GUI testing only:** You might not be able to run the Microsoft Visual Studio debugger if UFT is configured to record and run your test or component on any open Windows-based application. For tests, you can configure this by selecting **Run > Run Settings > Windows Applications**.

  Workaround: Do one of the following:

  - In the **MicIPC** section of the mercury.ini file (located in %SYSTEMROOT%) add these entries:

    ```
    devenv.exe=0
    msdev.exe=0
    msscrdbg.exe=0
    ```

  - Open the debugging program from a user account other than the one running UFT.

- **GUI testing only:** When Microsoft PDM 9.x or later is installed on your computer, if you add action automation objects to the **Debug > Watch** pane and then close and reopen your test without closing and opening UFT, these actions may not load successfully.

  Workaround: Restart UFT and open your test.

- If you are running a test from the ALM Test Lab module in **hidden mode** (as specified in the UFT Remote Agent, UFT will not stop the test at the breakpoints.

If you are running a test from the ALM Test Lab module not in hidden mode, the test stops at breakpoints if you select the **Run Test Sets in debug mode** option in the UFT Remote Agent.

For details on Remote Agent settings, see "Remote Agent Settings Dialog Box" on page 745.

- If you run a BPT test from the ALM Test Plan module or from UFT, UFT stops the test at all breakpoints in both **Debug** and **Normal** modes.

Part 5: UFT Integration With HP ALM

Chapter 25: ALM Integration

Relevant for: GUI tests and components and API testing

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

For a list of the supported versions of ALM or Quality Center, see the *HP Unified Functional Testing Product Availability Matrix*, available from the UFT Help or the root folder of the Unified Functional Testing DVD. The most up-to-date product availability matrix is available from the HP Software Product Manuals site, at http://h20230.www2.hp.com/selfsolve/manuals (requires an HP Passport).

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

This chapter includes:

Concepts

ALM Integration Overview

Relevant for: GUI tests and components and API testing

UFT integrates with ALM, the HP centralized quality solution. ALM helps you maintain a project of all kinds of tests (such as tests, components, business process tests, manual tests, tests created using other HP products, and so on) that cover all aspects of your application's functionality. Each test or component in your project is designed to fulfill a specified testing requirement of your application. To meet the goals of a project, you organize the tests in your project in unique groups.

ALM provides an intuitive and efficient method for scheduling and running tests or components, collecting results, analyzing the results, and managing test and component versions. It also features a system for tracking defects, enabling you to monitor defects closely from initial detection until resolution.

At its most basic level, integrating UFT with ALM enables you to store and access tests, components, application areas, and resource files in an ALM project, when UFT is connected to ALM.

For API testing users: UFT integrates with Service Test Management, ALM's extension for storing and managing application components, such as Web services. Service Test Management is an extension of ALM and it provides an efficient method for storing and retrieving application components.

This section also includes: "What is an ALM Project?" below

What is an ALM Project?

Relevant for: GUI tests and components and API testing

An ALM project is a database for collecting and storing data relevant to a testing process. For UFT to access an ALM project, you must connect to the local or remote Web server where ALM is installed.

When UFT is connected to ALM, you can:

- **Create tests, components, and resources and save them in your ALM project.**

- **View the contents of your tests.** This can help you decide if you want to run a test as part of a test set. Note that the Test Flow in ALM and the canvas in UFT display only the actions that are run when the currently selected test runs. This means that if a nested action is commented out, for example, that action is not displayed in ALM or in the UFT canvas. You can uncomment it in the UFT Editor when needed.

- **Run your tests and components and view the results in ALM**.

- **Associate a test, an API component, or a GUI component's application area with**

external files stored in the Test Resources module of an ALMproject.

- **Associate external files for all tests or for a single test.** For example, suppose you set the shared object repository mode as the default mode for new GUI tests. You can instruct UFT to use a specific object repository stored in the Test Resources module in ALM.Likewise, you can set the default activity repository location for your customAPI test activities in the Test Resources module in ALM.

  For GUI testing: For details on specifying external files for all new GUI tests, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544. For details on specifying external files for a single test, see "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.

  For API testing: For details on specifying the default location forAPI activity repositories, see "General Pane (Options Dialog Box > API Testing Tab)" on page 565.

- **Take advantage of all the features provided with the Resources and Dependencies model.** For details, see "Resources and Dependencies Model" on page 754.

GUI testing users can also use these additional features:

- **Use the QCUtil object to access and use the full functionality of the ALM OTA (Open Test Architecture).** This enables you to automate integration operations during a run session, such as reporting a defect directly to an ALM database. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing* and the ALM Open Test Architecture documentation.

- **Use the TDOTA object in your automation scripts to access the ALM OTA.** For details, see the *HP UFT Object Model Reference for GUI Testing* (**Help > HP UFT GUI Testing Advanced References Help > Automation Object Model Reference**).

- **Report defects to an ALM project either automatically as they occur, or manually directly from the Run Results Viewer.** For details on manually or automatically reporting defects to an ALM project, see and the sections on how to manually or automatically submit defects to an ALM project in the *HP Run Results Viewer User Guide*.

> **Tip:** You can save standalone, portable copies of tests stored in an ALM project when needed. For example, you may need to open or run a test while traveling because you do not have access to ALM. For details, see "Portable Copies of Tests" on page 133.

For details on working with ALM, see the *HP Application Lifecycle Management User Guide*.

For a list of required access permissions for working with ALM, see "Access Permissions " on page 73.

Template Tests

Relevant for: GUI tests only

Template tests serve as the basis for all tests created in ALM. A **template test** is a test that contains default test settings. For example, a template test might specify the UFT add-ins, associated function libraries, and recovery scenarios that are associated with a test. You can modify these test settings in the Test Settings dialog box (**File > Settings**) in UFT.

In addition to default test settings, a template test can also contain any comments or steps you want to include with all new tests created in ALM. For example, you may want to add a comment notifying users which add-ins are associated with the template test, or you may want to add a step that opens a specific Web page or application at the beginning of every test. Any steps or comments you add to a template test are included in all new tests created in ALM that are based on that template test.

All template tests are saved in your ALM project (except for the default template test, which is located on the ALM client) and do not need to be copied to each user's local computer. This enables users to customize their local default template tests, if needed, and still have access to globally maintained template tests.

When tests based on a specific template test are run from ALM, UFT automatically loads the associated add-ins and applies the required settings, as defined in the test.

Default Template Test

- A default template test is installed on each ALM client when the Unified Functional Testing Add-in for ALM is installed in the <Unified Functional Testing folder>\bin\Templates folder on your computer (for example: %ProgramFiles%\HP Software\Unified Functional Testing\bin\Templates\Template1).
- When an ALM user creates a new test in ALM, the default template test for the installed UFT version is automatically associated with the test unless the user selects another template test, as described in "How to Create a Template GUI Test" on page 730.
- The default template test is installed locally, therefore any changes you make to the template test are applied only to tests created on your computer (using the ALM client).

Working with New Template Tests

When you create new template tests, they are stored in your ALM project, making them available to all ALM users as the basis for new tests created in that ALM project.

You create a template test by first creating a blank test in UFT with the required test settings. Then, in the **Test Plan** module of your ALM project, you browse to your test and mark it as a **Template Test**. For details, see "How to Create a Template GUI Test" on page 730.

You can create multiple template tests, each for a specific testing purpose.

Example of Usage

You may want to create one template test for tests that test Web applications with ActiveX controls, and another for tests that test standard Windows application. You would associate the ActiveX and Web add-ins with the first template test. For the second template test, you would not associate any UFT add-ins at all, but you might specify the Windows application that you want to test. You could also make other modifications to the test settings for each of the template tests, as needed.

ALM Test Run Preferences

Relevant for: GUI tests only

You can run tests that are stored in an ALM database via:

- UFT
- ALM client that is installed on your computer
- A remote ALM client

ALM cannot run tests using UFT if the UFT computer is logged off or locked.

When ALM runs your test, it uses the associated add-ins list to load the proper add-ins for you test on the UFT computer. For details, see "Add-in Associations in a GUI Test" on page 583 and "Template Tests" on the previous page.

You can instruct UFT to report a defect for each failed step when an ALM test runs on your UFT computer. You can also submit defects to ALM manually from the Run Results Viewer. For details, see How to Manually Submit Defects to ALM (described in the *HP Run Results Viewer User Guide*).

Before an ALM client can run GUI tests on your computer, you must give ALM permission to use UFT (described in "How to Enable ALM to Run GUI Tests or Components on a UFT Computer" on page 726).

You can also view or modify the UFT Remote Agent settings (described in "How to View or Modify Remote Agent Settings" on page 734).

Hidden Mode

By default, UFT opens and runs in hidden mode when ALM activates it to run a test in a test set. This helps to improve performance.

You can change this setting in the UFT Remote Agent. You can also instruct the Remote Agent to display a tooltip window indicating that UFT is running an ALM test in hidden mode. For details, see "UFT Remote Agent Preferences" on page 715.

While UFT is running in hidden mode, you can open UFT to view the steps being run in the Editor by clicking the 🟦 button in the status bar.

Running Tests in Server-Side Execution

Relevant for: GUI tests and API testing

You can run tests in server-side execution when the tests are saved on a Lab Management-enabled ALM server.

Server-side execution enables ALM to run UFT tests on remote hosts at predefined times or on an ad-hoc basis, without anyone logged in to the host to initiate and control the test runs.

In contrast, client-side execution requires a user to be logged in to the host computer on which ALM runs the UFT test.

To set up server-side execution on ALM:

1. In the **Testing > Test Lab** module, create functional test sets. A functional test set is a group of automated tests or test configurations in an ALM project, designed to achieve a specific goal.

2. In the **Lab Resources > Testing Hosts** module, select hosts upon which the tests can run remotely.

3. If you want to use different values for specific parameters when running the tests on different environments or situations, you can define AUT parameters in the **Lab Resources > AUT Environments** module.

4. In the **Testing > Timeslots** module, schedule automatic test runs, or reserve timeslots to use for running manually.

   Tip: You do not need to reserve a timeslot if you run the tests ad-hoc.

For additional information, see the *HP Application Lifecycle Management User Guide*.

In UFT, you can link your test parameters to ALM AUT Environment parameters. This enables the test to use AUT environment parameter values passed from ALM when your UFT test runs in server-side execution. For details, see "AUT Environment Parameters" below.

Note: Server-side execution is available only for ALM Edition and Performance Center Edition, version 11.50 or later. You must also have Lab Management support for the ALM project.

For task details, see "How to Run a Test Using Server-Side Execution" on page 727.

AUT Environment Parameters

Relevant for: GUI tests and API testing

In ALM, you can define AUT environments, which contain AUT environment parameters, to be used in tests that run using server-side execution. When ALM runs the tests, it passes the values for the AUT parameters to the test.

To create multiple sets of parameter values for your test runs, you define multiple AUT environment configurations in your ALM project. This enables you to use different values for various data in your test, depending on the application, situation, or environment on which the test runs. For example, you may want to run the same test on different Web-based databases, each requiring a different URL, user name, and password.

In UFT, you can link test parameters to AUT parameters defined in a specific AUT environment in ALM. Subsequently, when ALM runs the test in server-side execution, the test parameters will receive the AUT environment parameter values passed from ALM.

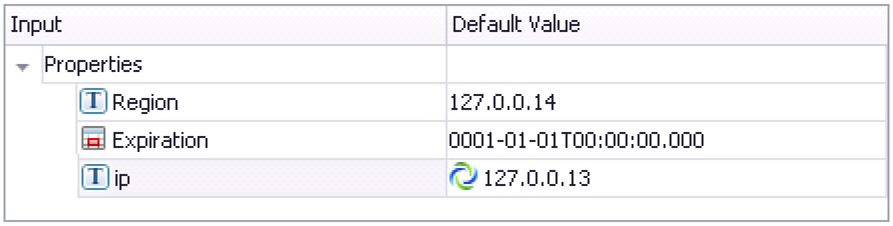
- When the test runs in server-side execution, it uses the values provided by ALM for the corresponding AUT environment parameter.
- When the test runs from UFT, the test uses the parameter’s default values.

For details on how to link test parameters to AUT parameters, see "How to Run a Test Using Server-Side Execution" on page 727

Note: All of the AUT parameters that you use must be from the same ALM AUT environment.

If you save the test to the file system, the links to the AUT parameters are removed, and the test parameters become ordinary parameters. The default values remain defined.

In the Properties pane, an ALM icon next to a parameter's default value indicates that it is an AUT parameter.



Input	Default Value
Properties	
Region	127.0.0.14
Expiration	0001-01-01T00:00:00.000
ip	127.0.0.13

What happens if the ALM AUT parameters linked to the UFT test parameters are deleted from ALM?

- If you open a GUI test with parameters linked to deleted AUT parameters, UFT displays a message specifying the missing parameters, and the AUT environment configuration to which they belonged. The the links to the missing AUT parameters are removed, and those test parameters become ordinary parameters.
- If you open an API test with test parameters that are linked to deleted ALM AUT parameters, the links remain unchanged.
- If you run a test with test parameters linked to deleted ALM AUT parameters, the test will fail to run.

UFT Remote Agent Preferences

Relevant for: GUI tests and components

When you run a test or business process test from ALM, the UFT Remote Agent opens on the UFT computer where the test runs. The UFT Remote Agent determines how UFT behaves when a test is run by a remote application such as ALM.

You can open the Remote Agent Settings dialog box at any time to view or modify the settings that UFT uses when ALM runs a test on your computer. For details, see "Remote Agent Settings Dialog Box" on page 745.

Data Awareness in ALM

Relevant for: GUI tests and components and API testing

> **Note:** The data awareness feature is available when integrating UFT with ALM. It requires the Unified Functional Testing Add-in for ALM or the QuickTest Professional Add-in for ALM.

In ALM, you can upload and store your Microsoft Excel (.xls) test data resource files in the Test Resources module, enabling you to use the same data for multiple tests or components, or different data for each test run.

In ALM, you run **configuration** instead of standard tests. When working with data-driven tests in ALM, each **configuration** is a test that is set to run with a selected data resource file and optional data filter settings. The filter settings enable you to filter data by specified row numbers, by parameter text values, or both.

> **Note:** You define and manage configurations only in ALM.

Example

> Suppose you define three configurations in a single test to test an application that provides different solutions for Gold Card, Silver card, and Bronze Card users. You could use the same data resource for each configuration by filtering the rows or text of the data resource to match the relevant user type. Similarly, if your data is stored in various .xls or .xlsx files, you might want to run the same test using a different data source each time. You would do this by associating a different data source with each configuration.

By managing your data as a resource in ALM, you can see at a glance which data resource files are associated with a particular test or configuration, and which tests or configurations are using a specific data resource. For more details, see "Advantages of Data Awareness in ALM" on page 717.

How Does Data Awareness in ALM Work?

When you create a test in ALM (or save a test to an ALM project from UFT for the first time), a default **configuration** with the same name as the test is created simultaneously. You can view this configuration in the Configuration tab of the Test Plan module.

In the ALM Test Resources module, you create one or more data resources and upload one Microsoft Excel (.xls or .xlsx) file for each.

After you upload a data resource file to your ALM project, you can define the test-level configuration settings for a particular test. You do this in the Parameters tab of the Test Plan module by selecting the data resource file and mapping its column names to the data table parameters in your test. You can associate one data resource with the test-level configuration. By mapping the column names to data table parameters, you enable UFT to identify and use the correct parameter value during a run session.

Note:

For GUI testing:

- Specifying a test-level data resource in the Parameters tab of the ALM Test Plan module overrides the **Data pane** settings in the Resources pane of the Test Settings dialog box.
- The data table parameters in your test must be defined in the **Global** sheet.

For API testing:

When running a test, API Testing's Data Navigation feature looks at the data resulting from the specific configuration. For example, if your ALM configuration filters the data to use only rows 3 through 5, and the API Test Navigation says to iterate rows 2 through 3, the actual data used will be rows 4 and 5 from the test resource.

In the configuration tab of the Test Plan module, you can create as many additional configurations as needed. By default, every configuration uses the test-level configuration settings unless you override the data resource.

If you associate a configuration with the data resource defined in the Parameters tab, you do not need to map the data table parameters to column names. You can, however, modify the filter settings by applying text filters to any parameter, and/or selecting the rows on which the configuration can run.

If you associate a different data resource with a configuration, or you select to override the test data resource and then select the same data resource file that you selected in the Parameters tab (which the same as selecting a different data resource), you need to map the data table parameters to the column names in the .xls or .xlsx file. You can also apply filter settings, as described previously.

If the data table parameter names in the **Global** sheet and the column names in the associated data resource are identical, you do not need to perform any mappings.

Next, in the Test Lab module, you can run any or all of the configurations defined for a test (or associated with a requirement) by adding them to a test set. When you run a configuration containing steps that use data table parameters, the parameter values are retrieved from the data resource files according to the settings defined for that configuration.

For a task describing how to work with configurations, see "How to Data Drive a Test Using Data from ALM" on page 720.

For a chart listing modules and tabs in which you perform tasks in ALM, see "ALM Data Awareness - Task Breakdown" on page 736.

This section also includes:

Advantages of Data Awareness in ALM

Relevant for: GUI tests and API testing

Test reuse. You can use the same test to test various scenarios, each time with a different set of data. You can do this associated a different data resource with each configuration, by associating a single data resource with different configurations and then filtering each configuration, or by using a combination of both.

Data Reuse. You can associate the same data resource with multiple tests or configurations.

Ease of Management. All of your resources are stored in one central location.

Visibility. You can see which tests are using a particular data table, and you can see which data table each test is using.

Requirements Coverage. You can cover all of your testing requirements by linking them to specific configurations. This enables you, for example, to use a single test to cover multiple requirements by associating different configurations in the same test with each requirement.

Resources and Dependencies Model. By storing your data in the Test Resources module, you can take advantage of additional ALM integration features, such as:

- Version control and baselines
- Asset comparison Tool and Asset Viewer
- Sharing data resources across ALM projects

Considerations for Data Awareness in ALM

Relevant for: GUI tests and API testing

Consider the following when managing your data resources in ALM:

- Only Microsoft Excel (.xls or .xlsx) files are supported as data resources.
- If you modify a data table parameter or a column name in the associated data resource after they are mapped to each other, you must update the mappings accordingly. Additionally, in API tests, if you update an Excel data source and link properties to the new data source, the new values will not be reflected in ALM until you remap the properties using the Map Parameters dialog box.

  For details, see "How to Data Drive a Test Using Data from ALM" on page 720.
- **For GUI testing only:** If the data table parameter names in your test's Global sheet are identical to the column names in the associated data resource file, you do not need to perform any mapping.

Tasks

How to Work with Tests and Components in ALM

Relevant for: GUI tests and components and API testing

The following steps describe the workflow of how to work with tests and components saved in an ALM project.

This task includes the following steps:

- "Prerequisites for Windows 7 and Windows Server 2008 R2 Users" below
- "Connect to an ALM Project" below
- "Create or open a test or component" on the next page
- "Create a template test (GUI testing only) - optional" on the next page
- "Create a test using a template test (GUI testing only) - optional" on the next page
- "Set UFT Remote Agent Preferences" on the next page
- "Run your test or component in the ALM project" on the next page
- "Manage versions of your project using version control - optional" on the next page
- "Disconnect from the ALM project" on the next page

Prerequisites for Windows 7 and Windows Server 2008 R2 Users

The security settings in Windows 7 and Windows Server 2008 R2 may prevent you from connecting to an ALM project.

This can occur when the UAC (User Account Control) option is set to ON, and you have never connected to an ALM project.

To connect to ALM for the first time, you must disable the UAC option and restart your computer. After you successfully connect to ALM, you can turn the UAC option on again. Thereafter, you should be able to connect to ALM, as needed. For details, see "Tools in the HP UFT program group" on page 2263.

Connect to an ALM Project

For details, see "ALM Connection Dialog Box" on page 737.

> **Note:** If you are connecting to an ALM server using external authentication, you are prompted as part of the login to select your external certificate.

Create or open a test or component

For details, see "New <Document> Dialog Box" on page 152 or "Open/New <Document>/<Resource> Dialog Box" on page 156.

Create a template test (GUI testing only) - optional

Perform this step to create a template test that has pre-defined test settings. You can then use this template test when creating new tests in ALM. For details, see "How to Create a Template GUI Test" on page 730.

Create a test using a template test (GUI testing only) - optional

Perform this step to create a test that has the pre-defined test settings defined in a template test. For details, see "How to Create a GUI Test in ALM Using a Template Test" on page 731.

Set UFT Remote Agent Preferences

Use the Remote Agent to view or modify the settings that UFT uses when ALM runs a test or business process test on your computer. For details, see "Remote Agent Settings Dialog Box" on page 745.

Run your test or component in the ALM project

For GUI testing:"How to Run a GUI Test, Business Process Test, or Component" on page 639 and "Run Dialog Box" on page 651

For API testing: "How to Run an API Test" on page 643 and "Run Dialog Box" on page 651.

Manage versions of your project using version control - optional

For details, see "Managing Versions of Assets in ALM Overview" on page 795.

Disconnect from the ALM project

For details, see "ALM Connection Dialog Box" on page 737.

> **Note:** When you disconnect from a project, all open documents from the project automatically close.

How to Data Drive a Test Using Data from ALM

Relevant for: GUI tests and API testing

This task provides a general overview of the steps involved in data driving a test with data stored in ALM. After you are familiar with the steps, you can perform many of them in the order you choose. Some steps may not be necessary in all cases.

> To see a demonstration of this functionality, go to the HP Live Network (HPLN) page and select the **HP ALM Data Awareness for UFT** movie from the list.

This task includes the following steps:

- "Prerequisites" below
- "Import data into a test (API testing only)" below
- "Data drive the test steps (API testing only)" below
- "Create a data resource file in your ALM project" below
- "Specify a default data table resource for all new test configurations" on the next page
- "Define your test configurations" on page 723
- "Link your configurations to requirements to create requirements coverage - Optional" on page 725
- "Run your test configuration" on page 725

1. Prerequisites

a. Connect to ALM, as described in "ALM Connection Dialog Box" on page 737.

b. Make sure that your test is saved in your ALM project. For details on creating tests and saving them to ALM, see "New <Document> Dialog Box" on page 152

c. **For GUI testing:** Make sure you have a test that uses data table parameters from the **Global** sheet. For details, see "How to Define Values for Your Step Arguments " on page 928.

2. Import data into a test (API testing only)

a. In the Data pane, click the **New Data Source** button and select **Excel**.

b. In the "New/Change Excel Data Source Dialog Box" (described on page 255), select the .xls or .xlsx file containing the data and Select the **Allow other tools to override the data** option. Click **OK** to import the data source into your test.

3. Data drive the test steps (API testing only)

For details, see "How to Assign Data to API Test/Component Steps" on page 1819.

4. Create a data resource file in your ALM project

In the Test Resources module:

a. Expand the Resources tree and select the required node.

b. Select **Resources > New Resource** to add a resource under that node.

c. In the New Resource dialog box:

   - In the **Type** list, select **Data table**.
   - In the **Name** box, enter a name for the data resource, for example, the name of the Microsoft Excel (.xls or .xlsx) file you plan to use.
   - Fill in the remaining fields (optional) and click **OK** to close the dialog box.

d. In the Resource Viewer tab, click **Upload File**. Then browse to and upload the relevant .xls or .xlsx file.

> **Note:** You can convert an internal data table from an open test to an uploadable data resource file by right-clicking the Data pane, selecting **File > Export**, saving the data table to the file system as an .xls or .xlsx file, and then uploading it as described above.

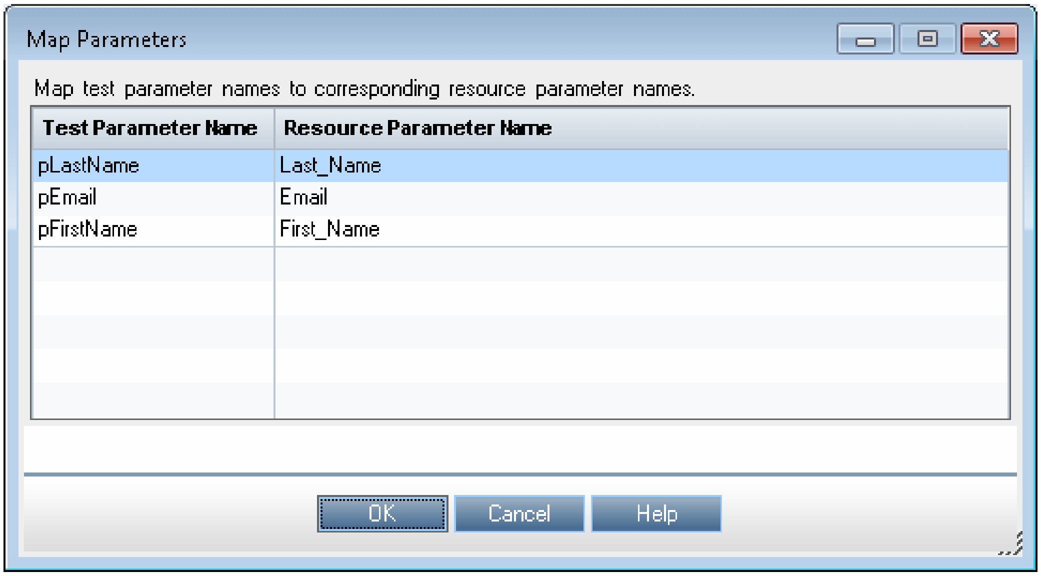
5. Specify a default data table resource for all new test configurations

a. In the Parameters tab of the Test Plan module, select the data table resource you want to use as the default for all test configurations.

   For GUI testing: If you do not specify a data table resource, the data specified in the Resources pane of the Test Settings dialog box (**File > Settings**) is used instead.

> **Note:** In this tab, only the Parameter Name column is relevant for GUI tests.

b. Click the **Map Parameters** button . In the Map Parameters dialog box, map the data table parameters (column headings) to the test parameters by entering the matching data table parameter names in the **Resource Parameter Name** column, as shown in the example below.



Map test parameter names to corresponding resource parameter names.

Test Parameter Name	Resource Parameter Name
pLastName	Last_Name
pEmail	Email
pFirstName	First_Name

All new configurations use the default mappings unless you specify otherwise in the Data tab of the Configurations tab. For details, see the next step, **Define your test configurations**.

6. Define your test configurations

Define test configurations for various run sessions. For each configuration, you specify whether to use the default resource file that you specified in the previous step, or whether to use a different data resource file.

a. In the ALM Test Plan module, browse to and select the test to which you want to associate your data table resource. For details, see the *HP Application Lifecycle Management User Guide*.

b. Click the **Test Configurations** tab. A default configuration is displayed in the grid. This configuration was created when your test was added to the ALM project. For details on configurations, see "Data Awareness in ALM" on page 715.

c. In the bottom pane of the Configurations tab, click the **Data** tab.

d. In the Data tab:

- Select the **Override test data resource** check box to select a different data resource file from the Test Resources module, or leave the check box blank to use the default resource file you selected in the Parameters tab in the previous step.

- In the **Data Resource** box, browse to and select the relevant data resource file to associate with this configuration. (Relevant only if you selected the **Override test data resource** check box)

- Click the **Data Resource Settings** button, and in the Filter Settings dialog box:

  - Map the data table parameters from your test to the column headers in the data table file (Relevant only if you selected a different data resource file in the previous step)

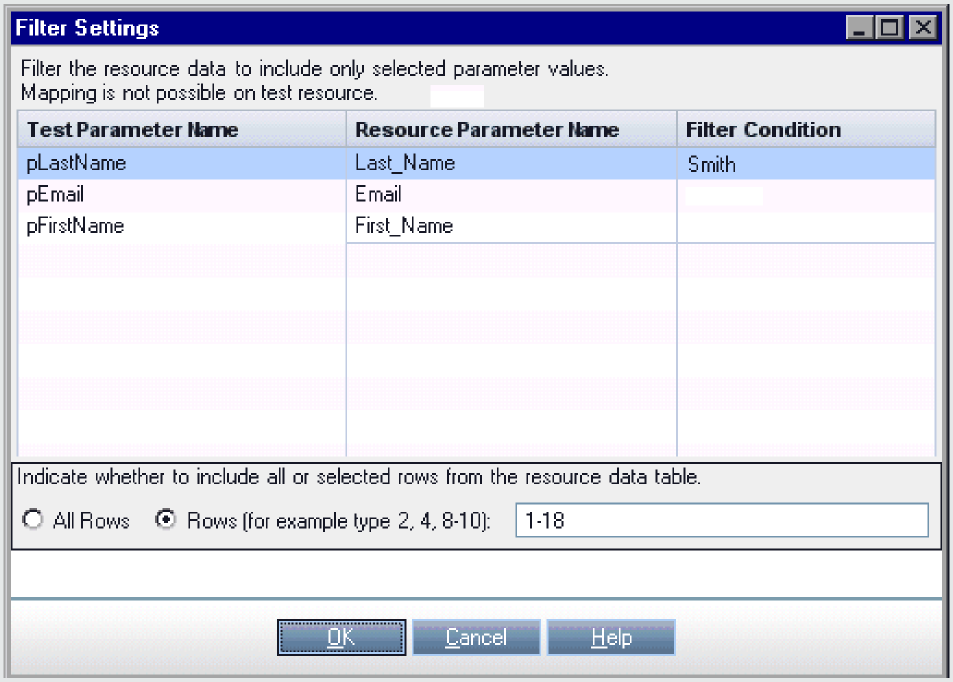
  - Apply filter conditions (text strings), as needed. You can apply one filter condition to each parameter.

  - Specify the rows on which to run iterations. For example, if you run a configuration named Gold, and users of this type are listed in rows 2-114, then specify these rows only.

> **Note:** If you apply filter conditions and specify rows, AND logic is used, meaning that the parameter value must equal the filter text value and the parameter value must be located in one of the specified rows.

> **Example:**

The pLastName data table parameter is mapped to Last_Name and is filtered to include only parameter text values that equal Smith. The configuration is set to run only on rows 1-18 in the .xls or .xlsx file you uploaded as the data resource.



Filter Settings

Filter the resource data to include only selected parameter values.
Mapping is not possible on test resource.

Test Parameter Name	Resource Parameter Name	Filter Condition
pLastName	Last_Name	Smith
pEmail	Email	
pFirstName	First_Name	

Indicate whether to include all or selected rows from the resource data table.

○ All Rows ● Rows (for example type 2, 4, 8-10): 1-18

For details on this dialog box, see the *HP Application Lifecycle Management User Guide*.

7. Link your configurations to requirements to create requirements coverage - Optional

If you want to make sure that your requirements are fully covered, link them to configurations. This enables you to select configurations to run based on requirement coverage when you plan your run session.

a. In the Test Plan module, click the **Req Coverage** tab.

b. Click the **Select Req** button. The Requirement Tree tab is displayed in the right pane.

c. From the Requirement Tree tab, select a requirement to add to the Req Coverage grid. When you add the requirement, the Add Advanced Coverage dialog box opens.

d. Select the test configurations that cover this requirement.

8. Run your test configuration

You run configurations from ALM.

a. **For GUI testing:** In UFT, make sure that in the **Tools > Options > GUITesting** tab **> Test Runs** node, the **Allow other HP products to run tests and components** is selected.

b. In the ALM Test Lab module, select or create a test set. For details, see the *HP Application Lifecycle Management User Guide*.

c. In the right pane, select the **Execution Grid** tab.

d. Click the **Select Tests** button to display the **Test Plan Tree** and **Requirements Tree** tabs in the right pane.

e. Do one of the following to select the configurations to run:

   - From the Test Plan Tree tab, select a test to add to the Execution Grid. When you add the test, all of its configurations are added to the Execution Grid. (The test itself is not added to the Execution Grid because ALM runs configurations, not tests.)

   - Below the Test Plan Tree tab, expand the **Test Configurations** pane, and add the specific configurations that you want to run to the Execution Grid.

   - Below the Requirements Tree tab, expand the coverage pane, and select a test to add to the Execution Grid. When you add the test, all of its configurations are added to the Execution Grid. (The test itself is not added to the Execution Grid because ALM runs configurations, not tests.)

f. Click the **Run** button to run the selected configurations.

g. After the run session, click the **Launch Report** button in the Last Run Report tab to view the results.

How to Enable ALM to Run GUI Tests or Components on a UFT Computer

Relevant for: GUI tests and components

To enable ALM to run tests or components:

In UFT, select the **Allow other HP products to run tests and components** option in the in the **Test Runs** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Test Runs** node). For details, see "Test Runs Pane (Options Dialog Box > GUI Testing Tab)" on page 539.

For security reasons, remote access to your UFT application is not enabled by default. This option enables ALM (or other remote access clients) to open and run tests.

To enable full access to tests from ALM:

Make sure that the Unified Functional Testing Add-in for ALM or BPT is installed on your ALM client computer. This enables you to view the test and view the run results in the Run Results Viewer. For details on this add-in, go to the Unified Functional Testing Add-in screen (accessible from the main ALM screen).

You can also optionally automatically upload your run results to ALM if you are running a test from ALM. This option is set in ALM as a site parameter for your project. For details, see the *HP Application Lifecycle Management Administrator Guide*.

For limitations for specific operating systems, see "Troubleshooting and Limitations - ALM Integration" on page 749.

How to Run a Test Using Server-Side Execution

Relevant for: GUI tests and API testing

This task describes how to run a UFT test from ALM, using server-side execution.

For an overview, see "Running Tests in Server-Side Execution" on page 713.

This task includes the following steps:

- "Prerequisites" below
- "Create tests and save them in ALM" below
- "Create functional test sets in ALM" below
- "Set up AUT Parameters in ALM and link your test parameters to them in UFT - optional" on the next page
- "Set up hosts in ALM on which the UFT tests can run " on page 729
- "Schedule the tests in ALM - optional" on page 729
- "Run the tests from ALM" on page 729

Most of the steps in this task must be performed on ALM. In UFT, you can link your test parameters to AUT parameters defined in ALM.

1. Prerequisites

- Connect to an ALM project with Lab Management support. For details on connecting to ALM, see "ALM Connection Dialog Box" on page 737.
- Make sure that HP ALM Lab Service is installed on the UFT computer.

For details, see the ALM Lab Management Guide.

2. Create tests and save them in ALM

Create your tests in UFT or ALM, and save them in ALM.

3. Create functional test sets in ALM

a. In ALM, in the **Testing > Test Lab** module, create a functional type test set and specify the required information.

b. Select the **Execution Grid** tab and click **Select Tests**.

c. In the Test Plan tree, select the tests you want to add to the test set.

4. **Set up AUT Parameters in ALM and link your test parameters to them in UFT - optional**

To use parameter values from value sets stored in ALM:

a. In ALM, in the **Lab Resources > AUT Environments** module, define AUT environments with parameters and set values for the parameters. For details, see the *HP Application Lifecycle Management User Guide*.

b. In UFT, open the Properties pane to the test parameters tab by doing one of the following:

- Open an API test, select the **Start** or **End** steps in the canvas, and open the Properties pane. Then select the Test Input/Output Parameters tab, described on page 417.

- Select a GUI test in the solution explorer. Open the Properties pane, and select the Parameters tab, described on page 394.

c. Click **Add > Add Input Parameter**.

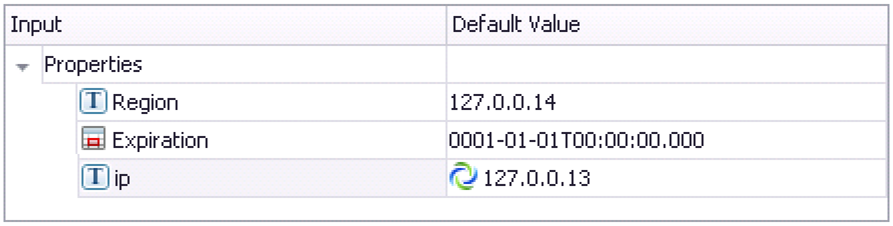
For details on the Add Input Parameter dialog box that opens, see "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)" on page 397 for GUI tests, or "Add Input/Output Property/Parameter Dialog Box (API Testing)" on page 1775 for API tests.

d. In the Add Input Parameter dialog box, click the **Select ALM application parameters** button.

e. In the Select AUT Parameter dialog box, expand the **AUT Environment** node and select a parameter. Click **OK**. For details, see "Select AUT Parameter Dialog Box" on page 742.

f. In the Add Input Parameter dialog box, provide a name for the parameter and set the default value, if necessary. Click **OK**.

In the Properties pane, an ALM icon next to a parameter's default value indicates that it is an AUT parameter.



Input	Default Value
Properties	
Region	127.0.0.14
Expiration	0001-01-01T00:00:00.000
ip	127.0.0.13

g. To edit the parameters, click the **Edit Parameter** button in the Properties pane.

h. To change the parameter to an ordinary parameter, without a linkage to an AUT environment, click the **Remove link to ALM application parameters** button ✖ in the Edit Parameter dialog box .

You can repeat these steps to add additional test parameters and link them to ALM AUT parameters, but all of the AUT parameters that you use must be from the same ALM AUT environment.

For more details, see "AUT Environment Parameters" on page 713.

5. Set up hosts in ALM on which the UFT tests can run

In ALM, in the **Lab Resources > Testing Hosts** module, you can view the hosts defined in the Lab Management project for all projects on your ALM server. Optionally, you can define additional host machines for your project.

When you define a new host, define its **purpose**, for example, QuickTest or Service Test, to indicate the type of test to run on this host. For details, see the *HP Application Lifecycle Management User Guide*.

Notes:

- Make sure that UFT is installed on the host computers you create or define in ALM.
- If UFT is not available in the list of purposes that you can select for a host, select **QuickTest Professional** for GUI tests or **Service Test** for API tests. Select both if you plan to run both types of tests.
- For each UFT host on which you want to run GUI tests, enable the **Allow other HP products to run tests and components** option in UFT. For details, see "Test Runs Pane (Options Dialog Box > GUI Testing Tab)" on page 539.

6. Schedule the tests in ALM - optional

In ALM, in the **Testing > Timeslots** module, schedule times for the tests to run automatically, or reserve timeslots to use later to run the tests manually.

If you are running the tests ad-hoc, you can skip this step.

7. Run the tests from ALM

In ALM's **Testing > Test Lab**, specify the hosts on which you want to run the tests, and run the tests. For details, see the *HP Application Lifecycle Management User Guide*.

You can also optionally automatically upload your run results to ALM if you are running a test from ALM. This option is set in ALM as a site parameter for your project. For details, see the *HP Application Lifecycle Management Administrator Guide*.

How to Use the ALM Connectivity Add-in

Relevant for: GUI tests and components and API testing

Install the ALM Connectivity Add-in

You integrate UFT with ALM using the ALM Connectivity Add-in. You can install this add-in as part of the installation..

You can also install it manually from the ALM Add-ins page by choosing **Help > Add-ins Page > HP ALM Connectivity** in ALM.

After you install the UFT Add-in for ALM as part of the standard installation, you must also install the Microsoft Visual C++ 2005 SP1 Redistributable Package on your computer. You can download this file from http://www.microsoft.com/en-us/download/details.aspx?id=5638.

View the version of the ALM Connectivity Add-in that is currently installed on your computer

Select **Help > About Unified Functional Testing** and select **Detailed Info.** For details, see "About Unified Functional Testing Dialog Box" on page 115.

How to Create a Template GUI Test

Relevant for: GUI tests only

This task describes how to create a template GUI test. For an overview, see "Template Tests" on page 711.

How to create a template GUI test in UFT

1. Open an existing test with the required add-ins loaded. For details on loading UFT add-ins, see the section on loading UFT add-ins in the *HP Unified Functional Testing Add-ins Guide*.

   > **Note:** Make sure that the add-ins included in the opened test are actually installed on the UFT computer on which the test will eventually run. Otherwise, when the test is run, UFT will not be able to load the required add-ins and the test may fail.

2. Define the required settings in the Test Settings dialog box (**File > Settings**). For details, see "Settings Overview" on page 581.

3. If you want to include comments or steps in all tests based on this template test, add them.

4. Select **File > Save As**. In the "Save <Resource>/Save <Document> As Dialog Box" (described on page 164), save the test to your ALM project using a descriptive name that clearly indicates its purpose.

   > **Tip:**

- In the ALM test plan tree (Test Plan module), you may want to create a special folder for your template tests. This enables other uses to quickly locate the relevant template test when they create new tests in ALM.
- When you save the test in UFT, you should apply a descriptive name that clearly indicates its purpose. For example, if the template test is used to associate the ActiveX and Web Add-ins with a new test, you could call it ActiveX_Web_Addins_Template.

How to create a template GUI test in ALM

1. Open the project in ALM, click the **Test Plan** button on the sidebar to open the Test Plan module, and browse to the test you previously created in UFT and saved in your ALM project.
2. Right-click the test and select **Mark as Template Test**. The test is converted to a template test.

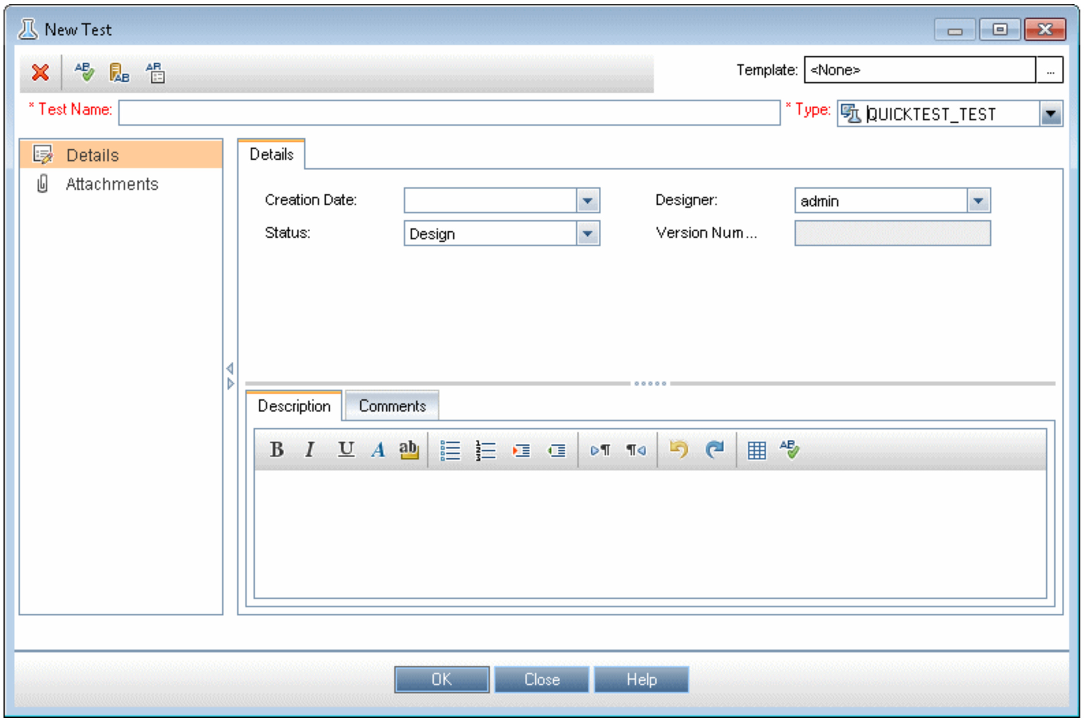
How to Create a GUI Test in ALM Using a Template Test

Relevant for: GUI tests only

This task describes how to create a GUI test in ALM using a template test as its basis.

1. In ALM, click the **Test Plan** button on the sidebar to open the Test Plan module.
2. In the test plan tree, select a folder.

3. Click the **New Test** button or select **Test > New Test**. The Create New Test dialog box opens.



> **Note:** The **Template** list is displayed only if the **Unified Functional Testing Add-in for ALM** is installed on your computer. If the **Template** box is not displayed, you must install the **Unified Functional Testing Add-in for ALM**from the Unified Functional Testing DVD or from the More ALM add-ins page (opened from the ALM Options window, or from **Help > Add-ins Page** in ALM.

4. From the **Test Type** list, select **QUICKTEST_TEST**.

5. In the **Test Name** box, type a name for the test start using English (Roman) letters, numbers, and underscores (if needed). For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269.

6. Click the **Template** box browse button. The Select Tests dialog box opens.

7. Expand the folder containing your template test.

8. Select the template test on which to base your new test and click the **Add** button . The Select Tests dialog box closes and the template test you selected is displayed in the **Template** box (in the Create New Test dialog box).

9. In the Create New Test dialog box, click **OK**. The new test is created with the test settings

defined in the template test.

10. The new test is displayed in the Test Plan tree under the subject folder you selected.

    Note: If the Required Fields dialog box opens, set the required values and click **OK**. For details, see the ALM administrator guides.

11. Continue creating the test. For details on creating tests in ALM, see the *HP Application Lifecycle Management User Guide*.

How to Enable the BPT Remote Agent

Relevant for: GUI tests and components and API testing

If the **Windows Firewall** is **turned on** on your computer, and you want to enable business process tests to be run on your computer from a remote ALM client, then you must manually create a firewall exception for the remote agent.

This task describes how to create a firewall exception for the Remote Agent.

1. Make sure you open the ALM or Quality Center client at least once on your computer.

2. Run **Firewall.cpl** from the command line. The Windows Firewall dialog box opens.

   Note: The remaining steps in this procedure may be different in different operating systems.

3. Click the **Exceptions** tab.

4. Click the **Add Program** button.

5. In the Add a Program dialog box, browse to the location where the ALM or Quality Center client is installed and select any of the following files that exist:

   - bp_exec_agent.exe
   - ComWrapperRemoteAgent.exe
   - BptRemoteAgenApplication.exe

   Note: You may have to repeat this step a few times to add all relevant files.

6. Click **OK** in the Add a Program dialog box. The files you selected are added to the Programs and Services list in the Windows Firewall dialog box.

7. Click **OK** to close the Windows Firewall dialog box.

For details about Windows firewall exceptions, see your documentation or contact your system administrator.

How to View or Modify Remote Agent Settings

Relevant for: GUI tests and components

This task describes how to view or modify the settings in the Remote Agent Settings dialog box.

1. Select **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Remote Agent**. The Remote Agent opens and the **Remote Agent** icon is displayed in the task bar tray.

   > **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

2. Right-click the **Remote Agent** icon and select **Settings**. The "Remote Agent Settings Dialog Box" dialog box (described on page 745) opens.

3. View or modify the settings in the dialog box. For details, see "Remote Agent Settings Dialog Box" on page 745.

4. Click **OK** to save your settings and close the dialog box.

5. Right-click the **Remote Agent** icon and select **Exit** to end the Remote Agent session.

How to Install a Certificate for ALM Servers in a Environment Using External Authentication

Relevant for: GUI tests and components, API testing and business process testing

This task describes how to install a certificate to use an ALM server running in a CAC (Common Access Card) environment.

1. Request the following from your certificate authority:

   - The certificate authority certificate in PEM format. Rename to **TrustedCA.pem**.

   - The web server certificate in PEM format. The full server name should appear in the certificate. Rename to **WebServerPublicCert.pem**.

   - The server certificate private key file in PEM format. Rename to **WebServerPrivateCert.pem**.

   - The software client certificate (if a Common Access Card is not used) for one user.

2. Place the certificate files in your web server configuration directory.

If you receive certificates in different formats, you can use **openssl** to convert them. To install openssl, go to http://www.openssl.org/related/binaries.html.

- To convert from CER, use **openssl x509 -in /<webserver-directory>/conf/cert.cer -outform pem -out cert.pem**.
- To convert from PFX, do the following:
  - Export the public key by using **openssl pkcs12 -in /<webserver-directory>/conf/cert.pfx -clcerts -nokeys -out certPublic.pem**.
  - Export the private key by using **openssl pkcs12 -in /<webserver-directory>/conf/cert.pfx -nocerts -nodes -out certPrivate.pem**.

Reference

ALM Data Awareness - Task Breakdown

Relevant for: GUI tests and components and API testing

The following table lists where to perform specific data awareness tasks in ALM:

Module	Tab	Task
Test Plan	**Parameters**	Define the test-level configuration for the current test: • Specify a data resource file. • Map the data table parameters used in your test steps to the column names in your data resource file.
	Test Configurations	Create additional configurations. (Optional) **Data tab:** • Select a different data resource. (Optional) • Specify the data resource settings: ▪ Specify the rows to be used during a run session. ▪ Specify any text parameters values filters. (Optional) ▪ If you select a different data resource (by overriding the data resource defined in the Parameters tab), map the data table parameters used in your test steps to the column names in your data resource file.
	Req Coverage	If you add requirement coverage to a test that contains multiple configurations, specify which configuration(s) to use for coverage. (Can also be done from the Requirements module)
Test Resources	**Resource Viewer**	• Create a data resource. • Upload a Microsoft Excel (.xls or .xlsx) file to use as the data resource.

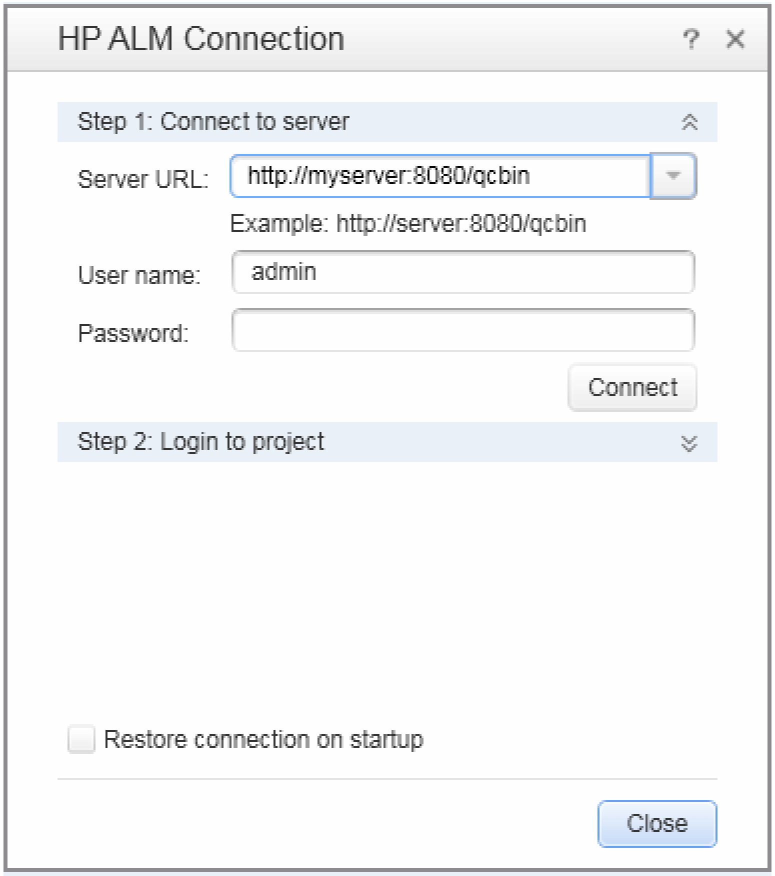
Module	Tab	Task
Test Lab	**Execution Grid**	• Add all of the configurations in a test to a test set. • Add only specific configurations to a test set. • Add all of the configurations contained in a tests that cover specific requirements to a test set. • Add only the configurations associated with a specific requirement to a test set.

ALM Connection Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to connect to or disconnect from a project in any supported version of ALM.

For more details, see "ALM Integration Overview" on page 709.



Server URL: http://myserver:8080/qcbin

User name: admin

Password:

□ Restore connection on startup

To access	Use one of the following: • **In UFT:** Select **ALM > ALM Connection**. • **In the Run Results Viewer:** Select **Tools > ALM Connection**. • Click the **ALM** toolbar button .

Important information	• **1st time connection.** The first time you connect to an ALM server, you must connect as a user with administrator privileges on the computer on which you are connecting. • **Connecting to ALM while a Web connection to Quality Center 10.00 is open.** You can simultaneously connect your Web browser to an ALM client and a Quality Center 10.00 client. However, if you want to connect to an ALM client from UFT, you must close any Web connections to a Quality Center 10.00 client. • **Connecting to different versions of Quality Center or ALM.** You cannot connect to multiple versions of Quality Center or ALM in the same UFT session. Close and reopen UFT to connect to a different version of Quality Center or ALM. • **Windows 7, Windows Server 2008 R2, Windows 8, and Windows Server 2012 Users.** The security settings in the following operating systems may prevent you from connecting to an ALM project: ▪ Windows 7 ▪ Windows Server 2008 R2 ▪ Windows 8 ▪ Windows Server 2012 This can occur when the UAC (User Account Control) option is set to ON, and you have never connected to an ALM project. To connect to ALM for the first time, you must disable the UAC option. After you successfully connect to ALM, you can turn the UAC option on again. Thereafter, you should be able to connect to ALM, as needed. For details see "Tools in the HP UFT program group" on page 2263. • **Connect.** The connection process has two steps. First, you connect to a local or remote ALM server. This server handles the connections between UFT and the ALM project. You must provide a user name and a password. Next, you choose the project you want to access. The project stores tests and run session information for the application you are testing. • **Disconnect.** You can disconnect from an ALM project or from an ALM server. ▪ If you disconnect from an ALM server without first disconnecting from a project, the UFT connection to that project database is automatically disconnected. ▪ After you open a Business Process test in UFT, you cannot log out of a

	project and log in to another project on the same server. You must first disconnect from the server and reconnect before logging in to another project. - **SSL Certificates.** If you are attempting to connect to an ALM project with a https:// prefix, but your computer does not have a valid SSL certificate, the connection will fail. - **ALM servers using external authentication.** To use an ALM server using external authentication, you must have your external certificate installed on both the computer running ALM and the computer running UFT. When you log in to the ALM server using external authentication, a dialog prompts you to select your certificate from the list of available certificates. - After running tests, from ALM, you can optionally automatically upload your run results to ALM if you are running a test from ALM. This option is set in ALM as a site parameter for your project. For details, see the *HP Application Lifecycle Management Administrator Guide*. If an ALM test or shared file (such as a shared object repository or data table file) is open when you disconnect from ALM, then UFT closes it in the document pane. The document remains in the Solution Explorer and is marked as unloaded.
Relevant tasks	To view the current connection: - **In UFT:** Point to the icon on the status bar. A tooltip displays the server name and project to which UFT is connected. - **In the Run Results Viewer:** The icon on the status bar is labeled with the server name and project to which the Run Results Viewer is connected.

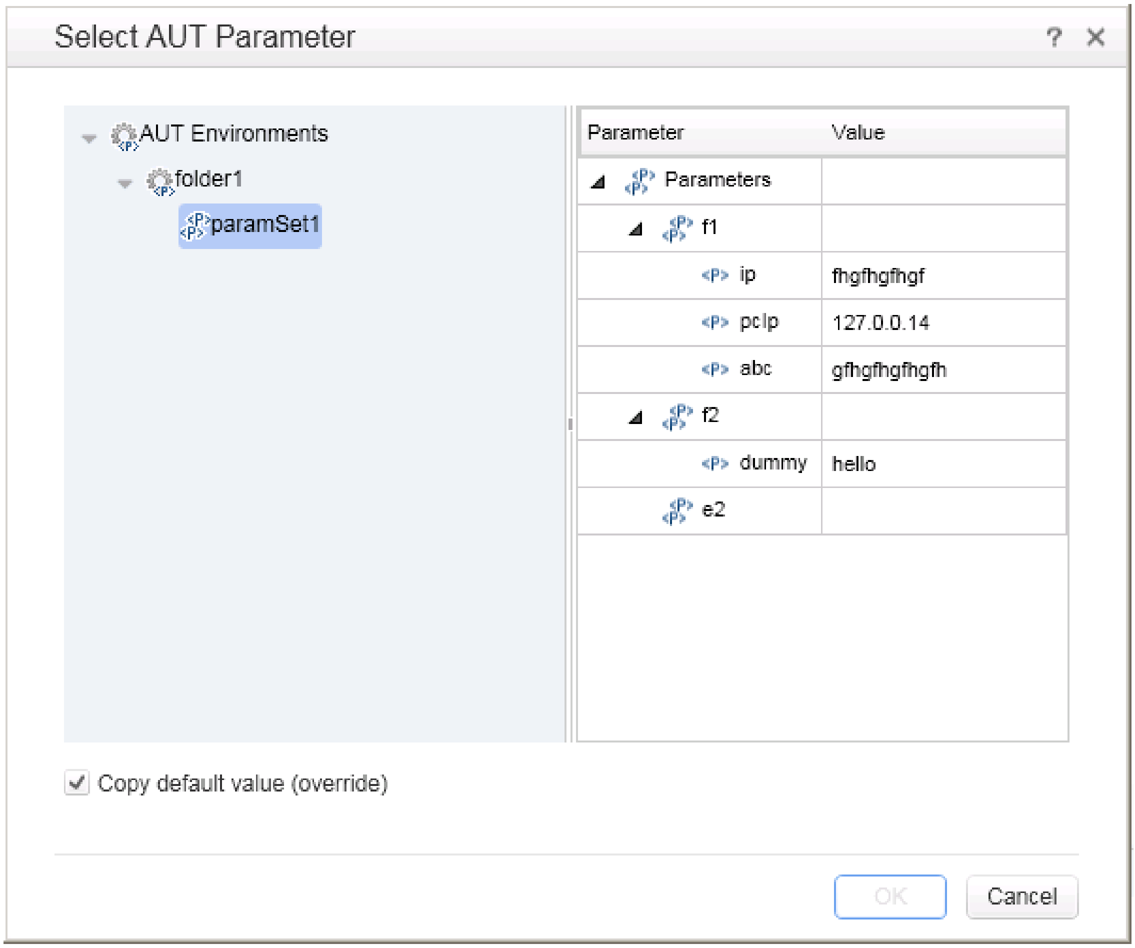
User interface elements are described below:

UI Elements	Description
Server URL	The URL address of the Web server where ALM is installed. You can choose a server that is accessible via a Local Area Network (LAN) or a Wide Area Network (WAN). You can connect to any currently supported version of ALM. For a list of supported versions, see the *HP Unified Functional Testing Product Availability MatrixHP Service Test Product Availability Matrix*, available from the root folder of the UFT DVD.
User Name	Your ALM user name. **Note:** If you are connecting to an ALM server using external authentication, you do not need to enter a user name. If you do enter a user name, UFT ignores the entered name when connecting.
Password	Your ALM password. **Note:** To enter a password in any CJK (Chinese, Japanese, Korean) language, copy/paste the password into the edit box. (Windows does not support typed CJK characters in password fields.) If you are connecting to an ALM server using external authentication, you do not need to enter a password. If you do enter a password, UFT ignores the entered password when connecting.
Connect / Disconnect	Connects to or disconnects from the selected ALM server. **Note:** After you successfully connect to a server, the button changes to **Disconnect**, and at the top of the dialog box, the **Disconnected** icon changes to a **Connected** icon.
Domain	The domain that contains the ALM project.
Project	The ALM project with which you want to work. **Note:** Only those projects for which you are a defined user are displayed.
Restore connection on startup	Instructs UFT to automatically reconnect to the ALM server every time you open UFT.

Select AUT Parameter Dialog Box

Relevant for: GUI tests and API testing

This dialog box enables you to link a test parameter to an AUT parameter contained in an ALM AUT environment configuration.



- AUT Environments
  - folder1
    - paramSet1

Parameter	Value
Parameters	
f1	
ip	fhgfhgfhgf
pcIp	127.0.0.14
abc	gfhgfhgfhgfh
f2	
dummy	hello
e2	

☑ Copy default value (override)

To access	1. Open a test stored on an ALM server with Lab Management enabled. Do one of the following: ▪ Open an API test, select the **Start** or **End** step in the canvas, and open the Properties pane. Then select the **Test Input/Output Parameters** tab . ▪ Select a GUI test in the solution explorer. Open the Properties pane, and select the Parameters tab . 2. Click **Add > Add Input Parameter** or select a parameter and click the **Edit Parameter** button . 3. In the Add/Edit Input Parameter dialog box, click . **Note:** Available only if UFT is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
Important information	AUT parameters are used for server-side execution. To set up server-side execution, create functional test sets in ALM, and use ALM's **Lab Resources** module to define hosts on which the tests run. For details, see "Running Tests in Server-Side Execution" on page 713, and the *HP Application Lifecycle Management User Guide*. **Note:** All of the AUT parameters that you use must be from the same ALM AUT environment.
Relevant tasks	"How to Run a Test Using Server-Side Execution" on page 727
See also	• "Add Input/Output Parameter for Test Settings" on page 1777 • "Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)" on page 397

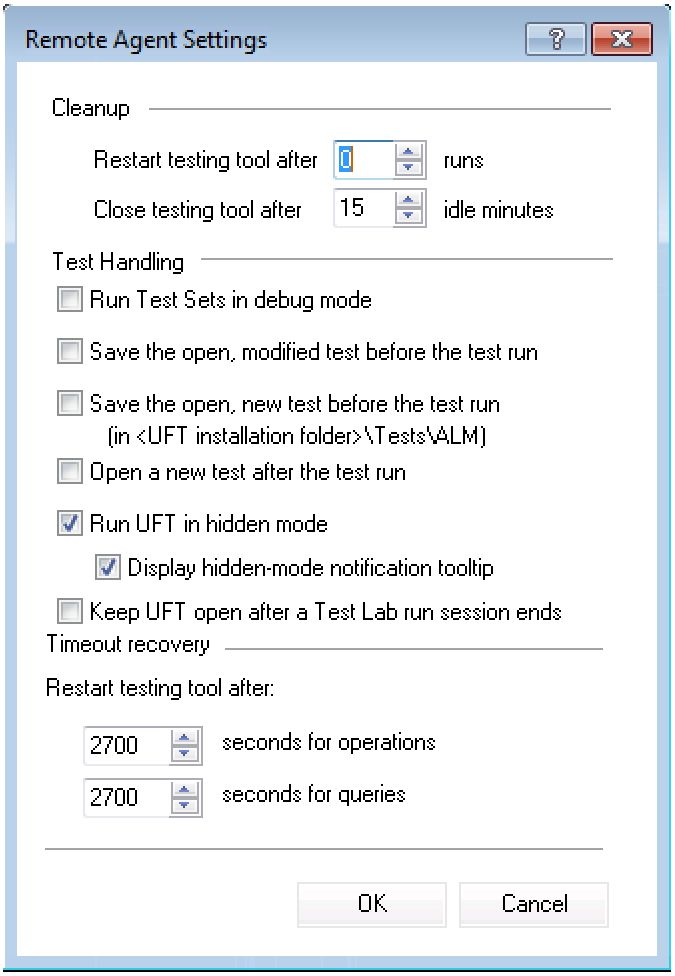
User interface elements are described below:

UI Elements	Description
AUT Environments	The list of AUT environment configurations defined in the **Lab Resources > AUT Environment** module. Expand the **AUT Environments** node to view the available AUT environments, and then select an environment to view the available parameters. If any test parameter is already linked to an AUT parameter, only the environment containing that AUT parameter is displayed.
Parameter column	The AUT parameter names, grouped in AUT environment configurations.
Value column	The value defined for the AUT parameter in ALM.
Copy default value (override)	Specifies whether to copy the parameter value displayed in this dialog box into the **Default Value** box in the Edit/Add Input Parameter dialog box. Copying overwrites any **Default Value** previously defined in the Edit/Add Input Parameter dialog box. However, after it is copied, you can edit the **Default Value** manually.

Remote Agent Settings Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to view or modify the settings that UFT uses when ALM runs a test or business process test on your computer.



Remote Agent Settings

Cleanup

Restart testing tool after 0 runs

Close testing tool after 15 idle minutes

Test Handling

□ Run Test Sets in debug mode

□ Save the open, modified test before the test run

□ Save the open, new test before the test run (in <UFT installation folder>\Tests\ALM)

□ Open a new test after the test run

☑ Run UFT in hidden mode

☑ Display hidden-mode notification tooltip

□ Keep UFT open after a Test Lab run session ends

Timeout recovery

Restart testing tool after:

2700 seconds for operations

2700 seconds for queries

To access	1. Select **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Remote Agent**. The Remote Agent opens and the **Remote Agent** icon is displayed in the task bar tray. 2. Right-click the **Remote Agent** icon and select **Settings**. The Remote Agent Settings dialog box opens. **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

Relevant tasks	"How to View or Modify Remote Agent Settings" on page 734.
See also	"UFT Remote Agent Preferences" on page 715.

User interface elements are described below:

UI Element	Description
Restart testing tool after __ runs	For tests, restarts UFT after ALM completes the specified number of tests runs. When UFT restarts, it continues with the next test in the test set. For business process tests, restarts UFT after ALM completes the specified number of component iterations. However, if it reaches the specified number of iterations in the middle of a business process test run, it waits until the current business process test iteration is finished before restarting. You may want to use this option to maximize available memory. If you do not want UFT to restart during a test set run, enter 0 (default).
Close testing tool after __ idle minutes	Closes UFT after it has been idle for the specified number of minutes.
Run GUITest Sets in debug mode GUI tests only	Instructs UFT to pause test runs at breakpoints inside in a test. **Note:** If you select this option when the **Run UFT in hidden mode** option is also selected, UFT does not stop at the breakpoints.
Save the open, modified test before the test run	If an existing test or keyword GUI component is open in UFT when the Remote Agent begins running a test, this option instructs UFT to save any unsaved changes to the open test or keyword component. **Note: (for GUI testing):** This option is not relevant for function libraries. Therefore, if an existing function library is open in UFT when the Remote Agent begins running a test, the function library is not saved.
Save the open, new test before the test run	If a new test is open in UFT when the Remote Agent begins running a test, the test is saved in:<Unified Functional Testing>\Tests\ALM with a sequential test name. **Note: (for GUI testing):** This option is relevant only for tests. Therefore, if a new keyword component or function library is open in UFT when the Remote Agent begins running a test, the component or function library is not saved.

UI Element	Description
Open a new test after the test run	By default, the last test or component run by the Remote Agent stays open in UFT when it finishes running the test or component. However, while it remains open, it is locked to other users. You can select this option to ensure that the last test or component that ALM runs on your computer is closed, and a blank test or component is open instead.
Run UFT in hidden mode	Specifies whether to run UFT in hidden (silent) mode when you run a test set from the Test Lab module in ALM. By default, this option is selected. **Display hidden-mode notification tooltip:** If this check box is selected, the Remote Agent displays a tooltip window when UFT runs an ALM test in hidden mode. You can click the tooltip to display UFT during the test set run. By default, this option is selected. **Note:** • Clicking the notification tooltip clears the **Run UFT in hidden mode** check box and UFT runs in normal mode. You can run UFT in hidden mode again by reselecting **Run UFT in hidden mode** before the next test set run. • When running in hidden mode, UFT can be optionally redisplayed at the end of each test or at the end of the test set. This behavior is configured in ALM Site Administration using the SUPPORT_TESTSET_END parameter. For details, see the section on setting ALM configuration parameters in the *HP Application Lifecycle Management Administrator Guide*. • If you are running a test from the ALM Test Lab module, UFT does not stop the test at any breakpoints inserted into the test.

UI Element	Description
Keep UFT open after a Test Lab run session ends	By default, when ALM opens UFT on a remote computer during a test set run (or when it runs selected tests or configurations from the Test Lab module), it closes UFT at the end of that Test Lab run session. This ensures that the UFT license is released at that point and made available for other UFT users. Selecting this option causes UFT to stay open on your computer (and to continue using the UFT license) after a Test Lab run session ends. **Note:** The behavior described above is relevant only when UFT is opened from an ALM server that has the SUPPORT_TESTSET_END parameter set to **Y**. (**Y** is the default setting). If UFT is opened from an earlier version of ALM (or the above-mentioned parameter is set to **N**), this option is ignored and UFT always remains open at the end of run session. For details on the SUPPORT_TESTEND_END parameter, see the section on setting ALM configuration parameters in the *HP Application Lifecycle Management Administrator Guide*.
Restart testing tool after	Restarts UFT if there is no response after the specified number of seconds for: • **Operations.**UFT operations such as Open or Run. • **Queries.** Standard status queries that remote applications perform to confirm that the application is responding (such as the **ALMget_status** query). The default value for both options is 2700 seconds (45 minutes). However, while UFT operations may take a long time between responses, queries usually take only several seconds. Therefore, you may want to set different values for each of these options. **Note:** If a function library with unsaved changes is open in UFT, UFT prompts you to save it. If the function library is not saved within 10 seconds, UFT restarts and any unsaved changes are lost.

Troubleshooting and Limitations - ALM Integration

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for working with tests and components and ALM, and includes:

Troubleshooting and Limitations - General ALM Integration

Relevant for: GUI tests and components and API testing

- The first time you connect to your ALM server, (either within UFT or through a browser) **you must connect as an administrator**. This allows the machine to properly install the ALM client with the required connectivity. For all subsequent connections, you do not need to log in as administrator. This step is also required after installing ALM patches.

- Before you run tests remotely from ALM, you must perform the following prerequisites:

  a. You must enable COM+ access. For details, see the Installation Guide.

  b. Change DCOM permissions and open firewall ports. For details, see the Installation Guide.

  c. Run RmtAgentFix.exe from the <Unified Functional Testing installation>\bin folder, or use the Additional Requirements Utility, which you open by selecting **Start > Programs > HP Software > HP Unified Functional Testing > Tools > Additional Installation Requirements**.

     This is due to a problem with opening DCOM permissions on Windows 7 and Windows Server 2008 R2.

  d. Disable User Account Control (UAC) in Windows and restart your computer before you first connect to ALM. For details, see "Troubleshooting and Limitations - UFT Program Management" on page 2263.

- The security settings in Windows 7, Windows Server 2008 R2, and Windows 2012 may prevent you from performing a UFT related installation, such as a patch installation, or connecting to an ALM project (either directly or from UFT). This can occur when the UAC (User Account Control) option is set to ON, and you have not yet connected to an ALM project (if relevant).

  Workaround: Temporarily turn off the UAC option. For details, see "Troubleshooting and Limitations - UFT Program Management" on page 2263.

  After disabling the UAC option as described above, perform the required installation or connect to ALM as usual. When you are finished, you can turn the User Account Control (UAC) option on again. Hereafter, you should be able to connect to ALM, as needed.

- In Windows 8, when trying connecting to ALM in UFT with the UAC turned off, you cannot connect to ALM.

  Workaround: Run UFT as an administrator (even if you are the admin user).

- If you are connected to an ALM server, and you want to connect to a different server, disconnect from the first ALM server, restart UFT, and connect to the second server.

- If UFT closes unexpectedly while an asset is open from ALM, the asset may remained locked by ALM for more than fifteen minutes. In some cases, you may be able to reopen UFT and reopen the test, but when trying to save it, you will receive an error message indicating that the test entity is already locked by you.

  Workaround: Wait fifteen minutes or more and try again.

- Renaming a test or component from ALM may cause the test or component to behave unexpectedly.

  Workaround: To rename a test or component, open it in UFT and rename it using the **Save As** option. If the test or component has already been renamed from ALM, use the rename option again to restore the old name, and then use the **Save As** option in UFT. Renaming a test parameter from UFT causes any run-time parameter values already set for this parameter in ALM to be lost.

- If an ALM user manually changes the status of a test instance run, ALM creates **fast run** results to record the change of the test status. The **fast run** results are not valid run results files. However, when you try to select results to open or delete in the Run Results Viewer or Run Results Deletion tool, the **fast run** results are available in the list.

- For tests or business components saved on ALM, if you perform a checkout from within ALM, it is not reflected in UFT for the current test—the test or business component still appears to be checked in.

  Workaround: Close and reopen the test in UFT.

- When running a test from UFT with ALM 11.50 and later with an ALM project that supports versioning, the test instance run status will not be updated for tests that are checked out.

- ALM supports non-Unicode projects. If you are working with an ALM project that is not Unicode compliant:

    - You should not use Unicode values (such as the name of the test or component, the name of an application area, the default value of a test, action, or component parameter, method argument values, and so forth).

    - Data that is sent to UFT from ALM (such as values for test, action, or component parameters) is not Unicode compliant.

    - UFT results containing Unicode characters may appear corrupted in the ALM result grid. You can, however, open and view results containing Unicode characters in the UFTRun Results Viewer.

      For additional details on UFT Unicode issues, see "Troubleshooting and Limitations - Multilingual Applications in GUI Testing" on page 2266.

- If there is a forward proxy with Basic Authentication between the server and client machines, before the first connection to an ALM platform, each ALM client must configure the proxy credentials by using the **Webgate Customization Tool**. If you do not run WebGate, you may be unable to connect, or you may need to enter your credentials multiple times.

  To run the tool:

  a. Go the following location on the ALM client machine: http:\\<ALM Platform server name>[<:port number>]/qcbin/Apps/

  b. Click on the **Webgate Customization Tool** link and select **Run**.

  c. In **WebGate Customization**, click in the **Proxy Credentials** area. Select the **Use these credentials** check box, and type values in the **Proxy Username** and **Proxy Password** boxes.

  d. Click **Save** and then **Close**.

- In UFT, when you run a test or flow that was recorded in QTP 11.00 using the BPTEE extension, and is currently stored in ALM 11.00, any ALM Detect Changes settings are ignored, and no changes are detected.

- In order to run a UFT test from an ALM server using external authentication, you must select the **Non-interactive mode** option in the Webgate Customization tool and select the external certificate in the same dialog.

Troubleshooting and Limitations - ALM Integration with GUI Testing

Relevant for: GUI tests and components

- If the parameter names in your **Global** data table are identical to the parameter names in an external data table, running a test configuration with data table parameter mapping to other data table parameters causes unexpected results. For example, if your **Global** data table parameters are Login and Password, and your external data table parameters are Login, Password, Login_localized, and Password_localized, a test configuration run with mapping from Login to Login_localized and Password to Password_localized will cause unexpected results.

  Workaround: Split the external data table into multiple tables.

- To run a test or component from ALM, you must first invoke UFT at least once. Otherwise, ALM may be unable to open UFT.

Troubleshooting and Limitations - ALM Integration with API Testing

Relevant for: API testing

- While connected to an ALM project, if you change the available license for an API test, ALM continues to use the original license.

  Workaround: Log out from your ALM project and restart Internet Explorer.

- For Service Test Management: Before importing a WSDL from Service Test Management, you must connect at least once to the Service Test Management-ALM server through Internet Explorer.

- To enable test versioning in Quality Center version 10.00, create a file in the <QC installation folder>\repository\sa\DomsInfo\Metadata\TEST folder called ServiceTest.xml with the following content:

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <Test Type="SERVICE-TEST">
    <Repository>
      <Folders ID="1" Filter="." BlindCopy="true" />
    </Repository>
    <Versioning Enabled="true" />
    <Baselining Enabled="false" />
   </Test>
  ```

- You can create and manage API components in much the same way as you would API tests, as described in "API Testing Design" on page 1588. For details about exceptions that apply when working with API components, see "Troubleshooting and Limitations - Business Process Testing in UFT" on page 2079.

- When editing a test on UFT on a computer running Windows XP, you must save your test before running it from ALM.

Chapter 26: Resources and Dependencies Model

Relevant for: GUI tests and components and API testing

> **Note:** The references to ALM in this chapter apply to Quality Center 10.00 and ALM. Note that some features and options may not be supported in the Quality Center or ALM edition you are using. For information on Quality Center or ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

Resources and Dependencies Model Overview

Relevant for: GUI tests and components and API testing

UFT enables you to use the Resources and Dependencies model to fully integrate your tests and components into ALM projects.

> **Note:** Before you read this section, you may want to familiarize yourself with the "Resources and Dependencies Model Terminology" (described on page 759).

- **Replaces the use of attachments with linked assets.** For example, **GUI tests, actions, and application areas** can be linked with function libraries and shared object repositories, respectively or **APItests** can be linked with data tables, user code files, or activities. You store your tests or components in the Test Plan or Business Components module, respectively, and you store your resource files (including application areas) in the Test Resources module. When you associate a resource file to a test or a GUI component's application area, these assets become linked. Linking assets improves runtime performance by decreasing download time. (Using attachments instead of resources increases download time from Quality Center 10.00 and ALM.) Linking assets also helps to ensure that the relationships between dependent assets are maintained.

- **Supports versioning for tests or components and resource files.** You can create versions of these assets in UFT or in ALM, and you can manage asset versions in ALM. For details, see "Version Control in ALM" on page 794.

- **Supports baselines for tests or components and resource files.** You can view baseline history in UFT, and you can view and manage baselines in ALM. For details, see "Baseline History Dialog Box" on page 805.

- **Enables you to view and compare your assets in both ALM and UFT**. You can use the Asset Comparison Tool to compare versions of individual assets and the Asset Viewer for viewing an earlier version of a asset. Both of these viewers are available in ALM and in UFT. For details, see "Viewing and Comparing Versions of UFT Assets" on page 770.

- **Enables you to import and share assets across ALM projects.** For details, see the *HP Application Lifecycle Management User Guide*.

For details about the advantages of working with this model, see "Asset Dependencies - Advantages" on the next page.

Asset Dependencies - Advantages

Relevant for: GUI tests and components and API testing

When you associate a **dependent** resource file with a test's or a GUI component's application area, the assets become integrally linked, and these links can be viewed in ALM (in the Dependencies tab of various modules) and—for external GUI actions—in UFT (in the Action Properties dialog box).

Assets stay linked

When you move a test, or rename a test, action, component, application area, folder, or resource, dependent assets are automatically updated to reflect the change. This helps ensure that there are no missing resources during a run session.

Resource files are all stored in one ALM module

Resource files are stored in the Test Resources module, enabling you to manage your resources in one central location, and to view at a glance which tests and application areas are using each resource file. For details on the Test Resources module, see the *HP Application Lifecycle Management User Guide*.

Using resources stored in the Test Resources module improves runtime performance

Tests or components open and run faster when the associated resource files are stored in the Test Resources module (instead of being stored as attachments to tests in the Test Plan module).

Version control can also be applied to resource files

When version control is enabled for a project, all of the assets can be checked into the version control database. For details, see "Version Control in ALM" on page 794.

You can create, view, compare, and run baselines

In ALM, you can create baselines that capture the developmental stage for each asset, view and compare these read-only baseline "snapshots", and run baselines from a project. In UFT, you can view and compare baselines. For details, see "Baseline History Dialog Box" on page 805.

You can share assets with other projects and synchronize them

You can copy assets from other projects. This enables you to reuse your existing assets instead of creating new assets whenever you create a new project. For example, you can create a "template" set of assets to use as a basis for new projects.

You can synchronize these assets in both projects when changes are made, or you can customize your assets to suit the unique needs of each development project. For details, see the sections on importing and synchronizing libraries in the *HP Application Lifecycle Management User Guide*.

Deleting assets is easier

When you delete an asset (a reusable GUI action or component or associated resource file), a warning message informs you if the asset is used by other tests (or more than once in the current test) or is associated with an application area. This message contains a **Details** section that lists the tests or application areas that are associated with this asset or contain calls to this action so you can modify the tests or application areas, as needed. This helps you manage your business process tests and GUI action calls so that tests do not inadvertently fail.

You can verify which tests or components are associated with specific resources and vice versa

In the ALM Test Plan module and Business Components module, you can highlight a test or component and, in the Dependencies tab, see which assets are using the test or component, and see which assets the test or component is using. Similarly, in the ALM Test Resources module, you can highlight a resource file and see the assets with which it is associated.

For details, see "Dependencies Tab (ALM Modules) " on page 762.

You can verify which tests contain calls to an action (GUI testing only)

You can view a list of the tests that contain calls to a particular action by focusing on the action and opening the Used By tab in the Action Properties dialog box. (Right-click an action in the canvas and select **Action Properties**.) For details, see "Used By Tab (Action Properties Dialog Box)" on page 909.

Reference

Relative Paths and ALM

Relevant for: GUI tests and components and API testing

Resource files and GUI actions that are associated with tests or application areas using a relative path are not considered dependencies. To ensure that your resource files are recognized as dependencies, they must be saved in the Test Resources module in ALM, and they must be associated using the full ALM path. This enables you to benefit from the features provided by the Resources and Dependencies Model, as described in "Asset Dependencies - Advantages" on page 756.

For GUI testing: Despite this, there may be cases in which you may want to use a relative path. For example, if your application is released in different languages, you may want to use a relative path when associating shared object repositories with your tests or component’s application areas, as this enables you to use the same test with localized shared object repositories.

Limitations - Relative Paths and ALM

- Run-time performance times are slower.
- Dependency information for these assets is not displayed in:
  - The Using and Used By grids in the Dependencies tab in ALM. For details, see: "Dependencies Tab (ALM Modules) " on page 762
  - The message box that opens when you delete an asset that is associated with other assets.For details, see "Asset Dependencies - Advantages" on page 756
  - **For GUI tests only:** The Used By tab of the Action Properties dialog box in UFT. For details, see: "Used By Tab (Action Properties Dialog Box)" on page 909
- ALM does not verify that these assets are included during the baseline verification process. For details, see "Viewing Baseline History" on page 798
- When opening or running tests or components from a baseline, any associated external GUI action or resource file that is associated via a relative path but is **not** included in the baseline is considered a missing resource. This may cause a test or component run to fail. (Note that the baseline version of an associated asset is used if the asset associated via a relative path **is** included in the baseline.), For details, see: "Viewing Baseline History" on page 798
- When using the Asset Comparison Tool to view a test or component, you cannot drill down to view assets that are associated via a relative path.
  See: "Asset Comparison Tool and Asset Viewer - Overview" on page 771

Resources and Dependencies Model Terminology

Relevant for: GUI tests and components and API testing

Term	Description
Asset	Any testing document or resource file, including: **For GUI testing:** • tests • actions (GUI testing only) • components • application areas (GUI testing only) • function libraries (GUI testing only) • shared object repositories (GUI testing only) • recovery scenarios (GUI testing only) • data table files • environment variable files **For API testing:** • tests • components • data table files • XML files • testing activities • Web references (done via Service Test Management) • user code files **Note:** In ALM, UFT assets are referred to by the more general term **entities**.

Term	Description
Resource	Any asset stored in the ALM Test Resources module that is used by a test or component. For example, a **GUI test or component** may contain calls to functions in associated function libraries, and it may reference test objects stored in shared object repositories that are associated with the test or component's application area. An **API test or component** may contain references to a data table file or use a custom testing activity. GUI testing resources include: • application areas • function libraries • shared object repositories • recovery scenarios • data table files • environment variable files API testing resources include: • data table files • XML files • testing activities • Web references (done via Service Test Management) • user code files **Note:** In some cases, a resource can be used by another resource. For example, a GUI test's recovery scenario can use functions in a function library.

Term	Description
Dependency	The linked relationship between a resource or external GUI action and a particular test or application area. Associated resource files and GUI actions are linked to each test or GUI component's application area that uses these resources or calls these GUI actions. In some cases, a resource can also be linked to another resource. For example, a GUI test's recovery scenario can call functions in a function library. Assets are considered dependencies if they are associated using absolute paths, and they are stored in the following modules: • **Test Plan module:** tests • **Business Components module:** components • **Test Resources module:** ▪ GUI testing resources like application areas, function libraries, shared object repositories, recovery scenarios, data table files, environment variable files ▪ API testing resources like data table files, .xml files, testing activities, and user code files **Note:** Tests or components stored in the **Unattached or Obsolete** folders in the Test Plan or Business Components module, respectively, are not considered dependencies because they are not associated with any test or component.
Configuration	A test that is set to run from ALM with an optional data resource file and optional data filter settings. In ALM, you can define various configurations for the same test or business process test. For tests, see "Data Awareness in ALM" on page 715 For components, see the *HP Business Process Testing User Guide*.

ALM Resources-Related User Interface

Relevant for: GUI tests and components and API testing

When you create an ALM project in your ALM server, the tests or components that you create in this project are saved to the Test Plan or Business Components module, respectively. You save your resource files to the Test Resources module. By associating resource files with your tests or GUI components' application areas, they become linked dependencies.

This section provides a general overview of the tabs that are relevant for tests, components, or applications areas. For details on using any of these tabs, see the relevant section in the *HP Application Lifecycle Management User Guide*.

This section includes (in alphabetical order):

Dependencies Tab (ALM Modules)

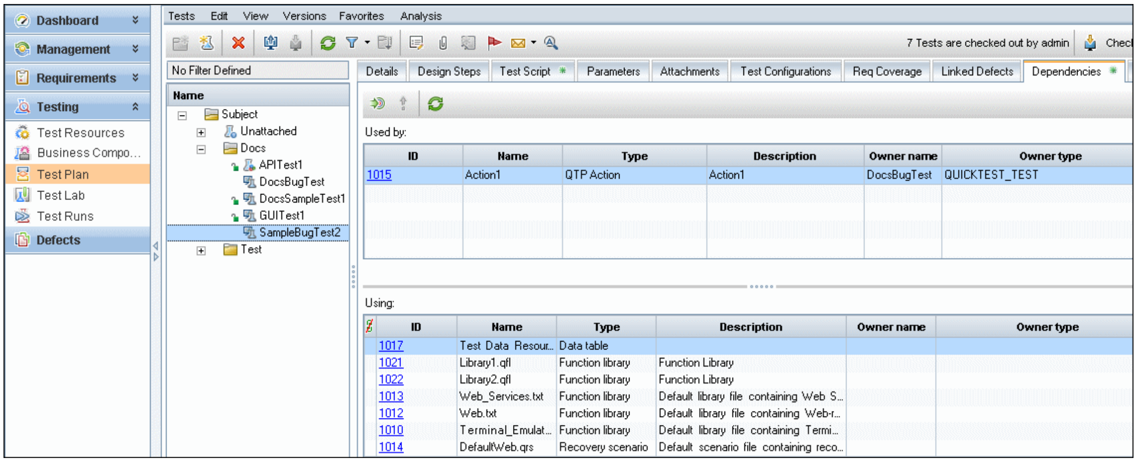
Relevant for: GUI tests and components and API testing

This tab displays the relationship between a selected asset, such as a test or component, and the assets with which it is associated. You use the Dependencies tab to see at a glance which resources are used by a particular asset, and which asset is using a particular resource. This information is displayed in the "Using Grid" on page 765 and "Used By Grid" on page 764 in the Dependencies tab in the Test Plan and Test Resources modules or the Business Components module.

Usage Example

For GUI testing: Suppose you want to modify the objects in a shared object repository. You can navigate to the shared object repository in the Test Resources module to view a list of the tests or application areas with which it is associated. This enables you to determine which assets this resource file is used by and helps you to analyze the impact that a proposed change may make to the dependent assets.

For API testing: Suppose you want to modify the values in a data table. You navigate to the data table in the Test Resources module to view a list of the tests or components with which it is associated. This enables you to determine which assets this resource file is used by and helps you to analyze the impact that a proposed change may make to the dependent assets.



Used by:

ID	Name	Type	Description	Owner name	Owner type
1015	Action1	QTP Action	Action1	DocsBugTest	QUICKTEST_TEST

Using:

ID	Name	Type	Description	Owner name	Owner type
1017	Test Data Resour...	Data table			
1021	Library1.qfl	Function library	Function Library		
1022	Library2.qfl	Function library	Function Library		
1013	Web_Services.txt	Function library	Default library file containing Web S...		
1012	Web.txt	Function library	Default library file containing Web-r...		
1010	Terminal_Emulat...	Function library	Default library file containing Termi...		
1014	DefaultWeb.qrs	Recovery scenario	Default scenario file containing reco...		

To access	In ALM, the Dependencies tab is available from the following modules: • Business Components module • Test Plan module • Test Resources module
Important information	For details on using this tab, see the *HP Application Lifecycle Management User Guide*. **For GUI testing:** When running a test with external resource files (like function libraries, data tables, recovery scenarios,etc.) that are saved in ALM, the resource files are not refreshed for each test run. As a result, any changes made to these external resource files during the current session are not reflected in a test run until you close and reload test and its resource files.
Relevant Tasks	• **For GUI testing:** In UFT, you can view **Used By** information for actions in the **Action Properties** dialog box. For details, see "Used By Tab (Action Properties Dialog Box)" on page 909. UFT also displays **Used By** information when you try to delete a dependent asset, so that you can determine how the change might affect associated assets before you delete the asset. • In the Business Components module, the Dependencies contains the following grids: ▪ **Used By.** Displays the business process tests that include this component. ▪ **Using.** Displays the application area with which the component is associated. For more details on the Dependencies tab in the Business Components module, see the *HP Business Process Testing User Guide*.
Business Process Testing	In the Business Components module, the Dependencies contains the following grids: • **Used By.** Displays the business process tests that include this component. • **Using.** Displays the application area with which the component is associated. For more details on the Dependencies tab in the Business Components module, see the *HP Business Process Testing User Guide*.

UFT-specific user interface elements are described below.

Used By Grid

This grid lists the assets that are using the asset currently selected in the tree. Suppose you are looking at the Used By grid for a shared object repository (for a GUI test) or a data table (for an API test). The Used By grid lists all of the GUI tests or application areas that are associated with the selected shared object repository or all the API tests that are associated with the data table. This list indicates the assets that will be affected if you modify or delete the asset selected in the tree.

Column	Description
ID	A unique numeric ID assigned automatically by ALM. If the **ID** is a link, you can click it to jump to that asset in ALM. **Example:** Suppose you are looking at the Used By grid for a specific function library (GUI testing only) or user code file (API testing only) in the Test Resources module. You can click the **ID** link to jump to the test or application area with which it is associated. (The link takes you to the Test Plan or Business Components module.)
Name	The name of the asset that is using the asset selected in the tree, for example, the name of a test, GUI action, application area, or business process test that is using the asset selected in the tree. **GUI-related** owner names include: • **Main Test Flow.** Indicates the test container called by the top-level action in the currently selected test in the Test Plan module. When **Main Test Flow** is displayed, the **Owner Type** is **Test**. • **Action<#>.** Indicates the internal name of the action that is called by an action in the currently selected test in the Test Plan module. **Action<#>** refers to the sequential number of the action when it was created. **Action<#>** is displayed when the **Owner Type** is **QTP Action**. **Note:Action<#>** is displayed even if an action was renamed in the test. • The actual name of the asset if the asset is not an action. An asset linked to an **API test or component** lists the name of the asset.
Type	The type of asset that is using the asset selected in the tree, for example ApplicationArea or BUSINESS-PROCESS or Test, QTP Action,QUICKTEST_TEST, or SERVICE_TEST.

Column	Description
Description	The description specified in the **Details** tab for the asset listed in the **Name** column (see description above). **Note: (for GUI testing)** • If the **Type** is **QTP Action**, displays the actual name of the action as shown in UFT (for example, if the action was renamed, the user-defined name is displayed) and displays its description, if any. • If the **Type** is **QUICKTEST_TEST** or **Test**, this cell is empty.
Owner Name (GUI tests only)	The name of the asset that owns the asset listed in the **Name** column. For example, if the asset listed in the **Name** column is an action, then the **Owner Name** is the name of the test in which that action is stored. (Not relevant for components or API tests)
Owner Type (GUI tests only)	The type of asset that owns the asset listed in the **Name** column. GUI-related owner types include: • **QUICKTEST_TEST.** A GUI test in the Test Plan module. • **QTP Action.** An action that is part of a test in the Test Plan module.

Using Grid

This grid lists all of the dependencies that the selected asset is using. For example, suppose you are looking at a test or component. You can see all of the external actions called by the test, all of the shared object repositories containing test objects used by the test or component, function libraries containing functions called by the test or component, and so on.

Column	Description
ID	A unique numeric ID assigned automatically by ALM.

Column	Description
Name	The name of the associated asset that the selected asset uses, for example, the name of the shared object repository, data table resource, or function library. GUI-related names include: • For **business process tests**, lists the names of the flows and/or components that are included in that test. • For **business process flows**, lists the name of the components that are included in that flow. • For **components**, lists the name of the associated application area. • **Action<#>.** Indicates the internal name of the action that is called by an action in a test in the Test Plan module. **Action<#>** refers to the sequential number of the action when it was created. **Action<#>** is displayed when the **Related Type** is **QTP Action**. **Note:Action<#>** is displayed even if an action was renamed in the test. • For **application areas**, lists the name of all associated resources, including function libraries and shared object repositories. • The actual name of the asset if the asset is not a test.
Type	The type of associated asset that the selected asset uses, for example, **ApplicationArea**, **Component**, **FLOW**, **QTP Action**, **Data table**, **Function library**, **Shared object repository**, and **Recovery scenario**.
Description	The description of the associated asset that the selected asset is using, if any. If the **Type** is **QTP Action**, displays the name of the action as shown in UFT and displays its description, if any.
Owner Name	The name of the asset that owns the asset listed in the **Name** column. For example, if the asset listed in the **Name** column is an action, then the **Owner Name** is the name of the test in which that action is stored. (Not relevant for components)
Owner Type	The type of asset that owns the asset listed in the **Name** column, for example, **QUICKTEST_TEST**. (Not relevant for components)

History Tab (ALM Modules)

Relevant for: GUI tests and components and API testing

This tab enables you to:

- View version information for a selected file.
- View baseline information for a selected file.
- View and compare file versions.
- View the baseline in which a version is stored (if applicable).
- Check out an earlier version of a file if you want to roll back to that version. (When you check the file back into the version control database, that version becomes the current version.)

To access	In ALM, the History tab is available from the following modules: • Business Components module • Test Plan module • Test Resources module **Note:** The History tab is located in the pane on the right side of the window. You may need to scroll to the right to display it.
Important information	For details on using this tab, see the *HP Application Lifecycle Management User Guide*.
Relevant tasks	In UFT, you can also view version history and baseline history by selecting one of the following: • **ALM > Version History** • **ALM > Baseline History**
See also	• "Version History Dialog Box" on page 803 • "Viewing Baseline History" on page 798

Libraries Module (ALM Modules)

Relevant for: GUI tests and components

This module enables you to:

- **Create, view, and compare baselines.** For details, see "Viewing Baseline History" on page 798 and the sections describing baselines in the *HP Application Lifecycle Management User Guide*.
- **Import assets from other ALM projects.** This enables you to create a complete copy of the assets that are included in a baseline in another project in any accessible domain. For details, see the *HP Application Lifecycle Management User Guide*.

To access	In the ALM sidebar, under **Management** select **Libraries**.
Important information	For details on using this tab, see the *HP Application Lifecycle Management User Guide*.

Troubleshooting and Limitations - Resources and Dependencies

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for resources and dependencies.

- When you save a resource to ALM (either from UFT or using the **Upload** option from the ALM Test Resources module), and the resource file has a comma in the file name, the resource appears to be saved successfully, but the file is not actually uploaded to the ALM server.

- **For GUI testing:** If you insert a call to an external action that is associated with a data table, and that data table was previously renamed or moved in the Test Resources module of Quality Center 10.00 or ALM, UFT tries to locate the data table in its original location.

  Workaround: Save the test, close it, and reopen it.

Chapter 27: Viewing and Comparing Versions of UFT Assets

Relevant for: GUI tests and components and API testing

> **Note:** The references to ALM in this chapter apply to Quality Center 10.00 and ALM. Note that some features and options may not be supported in the Quality Center or ALM edition you are using. For information on Quality Center or ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

Asset Comparison Tool and Asset Viewer - Overview

Relevant for: GUI tests and components and API testing

An **asset** is a UFT testing document (such as a test, component, or application area) or any resource file that is used by a UFT testing document (such as a function library, a shared object repository, a data table, a recovery scenario, or an environment variable XML file). The Asset Comparison Tool and Asset Viewer enable you to view and compare versions of a particular asset.

Using these tools, you can:

View any version of an asset using the Asset Viewer

For details, see "Asset Viewer" on page 788.

Compare two versions of an asset using the Asset Comparison Tool

For details, see "Asset Comparison Tool" on page 780.

Drill down to view or compare versions

Drilling down enables you to:

- View or compare versions of an **integral element**. An integral element is a resource file that is a part of the test or component (not saved as an external resource), such as a local object repository (for GUI testing) or a data table (for API testing). When you check in a test or component, these elements are checked in, too, because they are part of the test or component. Therefore, when you drill down in the asset, you can view or compare the version that existed when the test or component was checked in, in addition to the currently saved version.

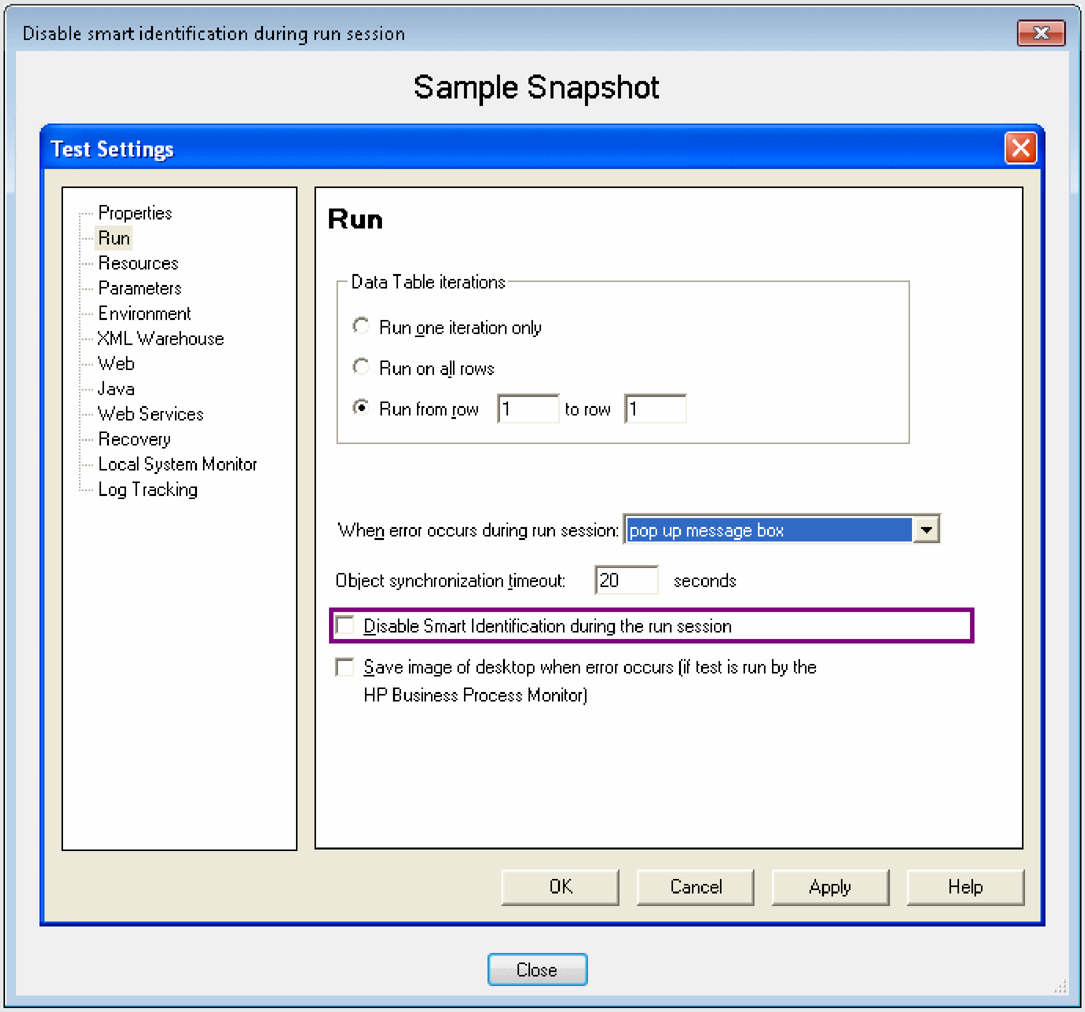
- View or compare versions of **associated external assets.** An associated asset is any external (not integral) resource file used by an asset (such as a function library, a shared object repository, a data table, a recovery scenario, or an environment variable XML file).

> **Note:**
>
> - When you drill down while viewing or comparing an asset, the currently checked in content of the associated assets is displayed, even if you are viewing or comparing an earlier version of the main asset.
>
>   To view or compare an earlier version of the drilled-down associated asset, open the resource file itself and use the Asset Viewer or Asset Comparison Tool.
>
> - To view or compare by drilling down, the resource file must be associated via an absolute or ALM path. For details, see "Relative Paths and ALM" on page 758.

View a screen capture depicting the UFT location of an element (GUI testing only)

The screen capture displays an example of the relevant dialog box. The option (or area) for the node you right-clicked is highlighted in the screen capture.

Example for tests: Suppose you are viewing a comparison of a test and you notice that the Disable Smart Identification during the run session node is highlighted, indicating that it was modified. If you are not sure where this option is located in UFT, you can right-click the node in the comparison tree and select **View Sample Snapshot**. UFT then opens a dialog box showing you that this area is located in the Run pane of the Test Settings dialog box. The title bar of the dialog box lists the selected element, and a highlighted rectangle outlines the option.



Disable smart identification during run session

Sample Snapshot

Test Settings

Run

Data Table iterations

○ Run one iteration only

○ Run on all rows

● Run from row 1 to row 1

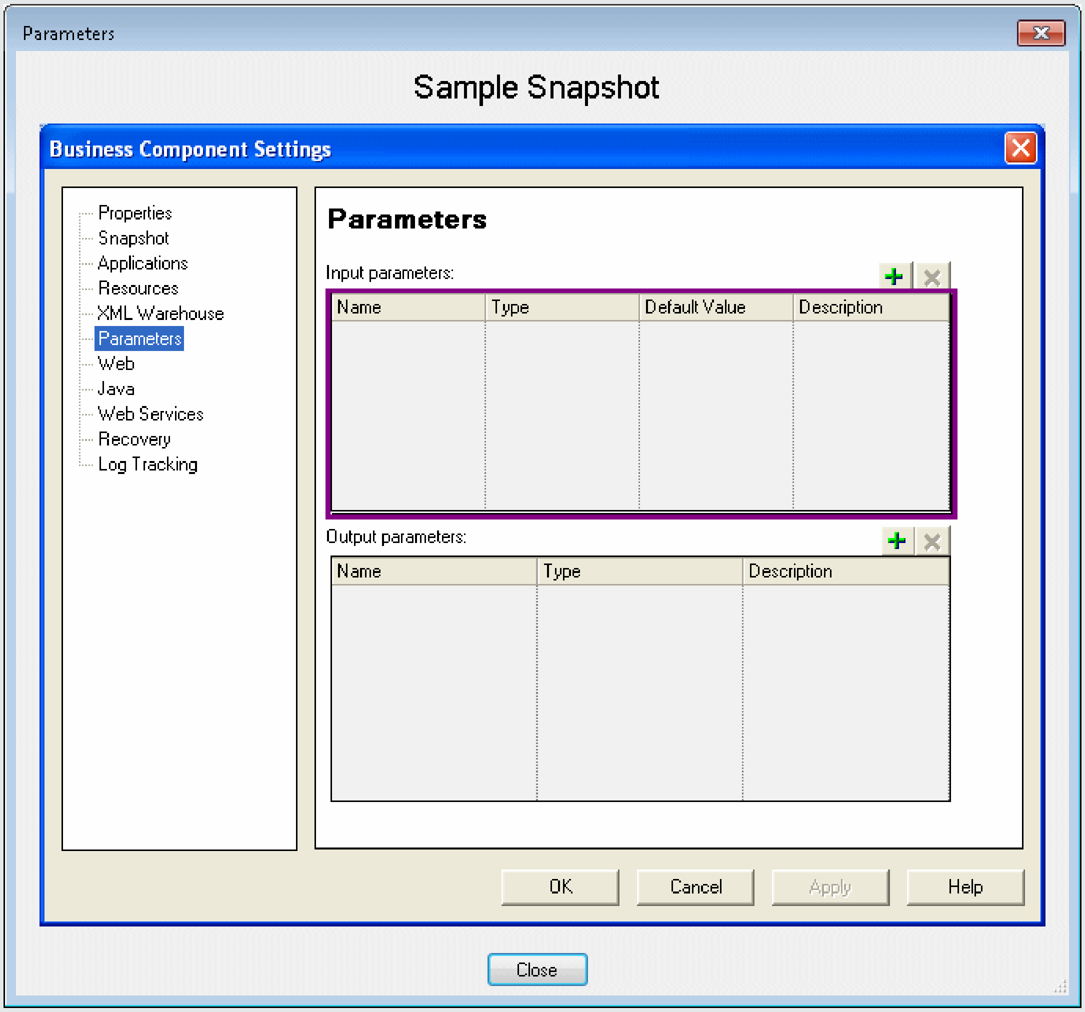
When error occurs during run session: pop up message box

Object synchronization timeout: 20 seconds

□ Disable Smart Identification during the run session

□ Save image of desktop when error occurs (if test is run by the HP Business Process Monitor)

Example for components: Suppose you are viewing a comparison of a component and you notice that the Input parameters node is highlighted, indicating that it was modified. If you are not sure of where this option is located in UFT, you can right-click the node in the comparison tree and select **View Sample Snapshot**. UFT then opens a dialog box showing you that this area is located in the Parameters pane of the Business Component Settings dialog box. The title bar of the dialog box lists the selected element, and a highlighted rectangle outlines the option.



For details, see "How to Work with the Asset Comparison Tool and Asset Viewer" on page 778 and "Shortcut Menu Commands (GUI testing only)" on page 784.

Tasks

How to Open the Asset Comparison Tool

Relevant for: GUI tests and components and API testing

This task describes how to open the Asset Comparison Tool.

Prerequisites

To open the Asset Comparison Tool from UFT or from ALM, the asset must be saved in a version control-enabled ALM project or in a baseline of an ALM project.

Open the Asset Comparison Tool from any of the following:

The main UFT window

1. Open the test, component, or function library (GUI testing only) whose versions you want to compare.

2. Select **ALM > Version History** or **Baseline History**. The Version History or Baseline History dialog box opens.

3. Select two versions (using the CTRL key) and click **Compare**. The Asset Comparison Tool opens.

The Object Repository Manager (GUI testing only)

1. Open the Object Repository Manager (**Resources > Object Repository Manager**).

2. Browse to and open the shared object repository whose versions you want to compare. For details, see "Open a shared object repository" on page 1292.

3. Select **ALM > Version History** or **Baseline History**. The Version History or Baseline History dialog box opens.

4. Select two versions (using the CTRL key) and click **Compare**. The Asset Comparison Tool opens.

The Recovery Scenario Manager (GUI testing only)

1. Open the Recovery Scenario Manager (**Resources > Recovery Scenario Manager**).

2. Open the recovery scenario file whose versions you want to compare. For details, see "Recovery Scenario Manager Dialog Box" on page 1121.

3. Click the **Version Control** down arrow and select **Version History** or **Baseline History**.

4. Select two versions (using the CTRL key) and click **Compare**. The Asset Comparison Tool opens.

From within ALM

1. In ALM, connect to the project containing the asset you want to compare.

2. Do one of the following:

   - In the sidebar, click the **Test Plan** button (for tests) or the **Business Components** button (for components) to open the Test Plan or Business Components module.

   - Click the **Test Resources** button in the sidebar to open the **Test Resources** module. This module contains the resource files associated with your test or component, such as function libraries, shared object repositories, data tables, recovery scenarios, or environment variable XML files.

3. In the tree, select the file whose versions you want to compare.

4. Click the **History** tab, and then click the **Versions** or **Baselines** tab.

5. In the grid, select two versions to compare (using the CTRL key), and then click the **Compare** button.

6. In the sidebar of the window that opens, click the **QTP Comparison/ST Comparison** or **Automation** button. The Asset Comparison Tool opens.

   > **Tip:** You can also compare baselines from the **Management** module. Click the **Management** button in the side bar to open the **Management** module. Select a baseline in the tree and click the **Compare To** button. For details, see the ALM user guide. For more details on baselines, see "Managing Versions of Assets in ALM Overview" on page 795.

The Command Line Interpreter (cmd.exe) (tests only)

You can use the UFTDiffApplication.exe tool from the command line to compare two assets from the file system.

1. Open the Command Line Interpreter.

2. Enter the command using the following syntax:

   ```
   "<UFT installation folder>\bin\UFTDiffApplication.exe" P1: "<file path 1>" P2: "<file path 2>"
   ```

   where **P1** = the file system path to the first asset, and **P2** = the file system path to the second asset.

> **Note:** Make sure you insert a blank space after each argument. The options are not case-sensitive and can be entered in any order.

> **Example**
>
> ""%ProgramFiles%\HP\Unified Functional Testing\bin\UFTDiffApplication.exe"
> P1: "%ProgramFiles%\HP\Unified Functional Testing\Tests\Test1"
> P2: "%ProgramFiles%\HP\Unified Functional Testing\Tests\Test2"

How to Open the Asset Viewer

Relevant for: GUI tests and components and API testing

This task describes how to open the Asset Viewer.

Prerequisites

To open the Asset Viewer from UFT or from ALM, the asset must be saved in a version control-enabled ALM project.

Open the Asset Viewer from any of the following:

The main UFT window

1. Open the test, component, or function library for which you want to view an earlier version.
2. Select **ALM > Version History**. The Version History dialog box opens.
3. Select a version and click **View**. The Asset Viewer opens.

The Object Repository Manager (GUI testing only)

1. Open the Object Repository Manager (**Resources > Object Repository Manager**).
2. Browse to and open the shared object repository for which you want to view an earlier version. For details, see "Open a shared object repository" on page 1292.
3. Select **ALM > Version History**. The Version History dialog box opens.
4. Select a version and click **View**. The Asset Viewer opens.

The Recovery Scenario Manager (GUI testing only)

1. Open the Recovery Scenario Manager (**Resources > Recovery Scenario Manager**).
2. Open the recovery scenario file for which you want to view an earlier version. For details, see "Recovery Scenario Manager Dialog Box" on page 1121.

3. Click the **Version Control** down arrow and select **Version History**.

4. Select a version and click **View**. The Asset Viewer opens.

From within ALM

Connect to the project containing the asset you want to view, and do one of the following:

To view the current version of an asset:

In the Test Resources module, select the resource and click the **Resource Viewer** tab.

To view the current or earlier version of an asset:

1. Do one of the following:

   - In the sidebar, click the **Test Plan** button (for tests) or the **Business Components** button (for components) to open the Test Plan or Business Components module.

   - Click the **Test Resources** button to open the **Test Resources** module. This module contains the resource files associated with your test or component, such as function libraries, shared object repositories, data tables, recovery scenarios, and environment variable XML files.

2. In the tree, select the file for which you want to view an earlier version.

3. Click the **History** tab, and then click the **Versions** or **Baselines** tab.

4. In the grid, select a version, and then click the **View** button. (You cannot view a version that is currently checked out.) A window opens with buttons in the sidebar enabling you to access version-specific information for the selected asset. (These buttons are identical to the tabs displayed in the right pane of the main window for the latest version of the selected asset.) For details, see the ALM user guide.

The Command Line Interpreter (cmd.exe) (tests only)

You can use the UFTDiffApplication.exe tool from the command line to view an asset from the file system.

1. Open the Command Line Interpreter.

2. Enter the command using the following syntax:

   ```
   "<UFT installation folder>\bin\UFTDiffApplication.exe" P1: "<file path 1>"
   ```

   where **P1** = the file system path to the asset.

   Example:

```
"%ProgramFiles%\HP\Unified Functional Testing\bin\UFTDiffApplication.exe"
P1: "%ProgramFiles%\HP\Unified Functional Testing\Tests\Test1"
```

How to Work with the Asset Comparison Tool and Asset Viewer

Relevant for: GUI tests and components and API testing

The following steps describe the tasks most often performed using the Asset Comparison Tool and the Asset Viewer.

View a comparison of two asset versions (Asset Comparison Tool)

For details, see "Asset Comparison Tool" on page 780.

Drill down to compare or view a specific element (GUI testing only)

> **Note:** You can drill down any asset that has a blue drilldown arrow ↓ adjacent to it.

- Click the blue drilldown arrow ↓ adjacent to any asset that can be compared. (The pointer changes into a pointing hand in the proximity of the drilldown arrow.)

- Double-click the asset.

- Right-click the asset and select **View Drilldown**. For details, see "Shortcut Menu Commands (GUI testing only)" on page 784.

- Select an asset and press **ENTER** on your keyboard.

View the UFT location of an element (GUI testing only)

Right-click the relevant node and select **View Sample Snapshot**. The screen capture displays an example of the relevant dialog box. The option (or area) for the node you right-clicked is highlighted in the screen capture. For details, see "View a screen capture depicting the UFT location of an element (GUI testing only)" on page 772.

Modify text and background colors

Modify the text and background colors for the filter types (modified, added, deleted, and so on) in the Asset Comparison Tool window using the Color Settings dialog box (described on page 785).

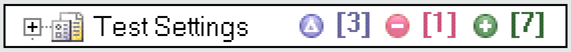
When you modify the background color of a filter type, the color of the filter type in the legend at the top of the window changes accordingly. These changes remain in effect unless you change them again or restore the default settings.

View the number of differences for a specific element

In the Asset Comparison Tool, collapse the node representing an element.

If the sub-elements of that element are different between versions, a legend is displayed adjacent to the node. The legend indicates the number of differences that exist under the collapsed element.

In the following example, three sub-elements were modified, one was deleted, and seven were added:



For details, see "Legend" on page 784.

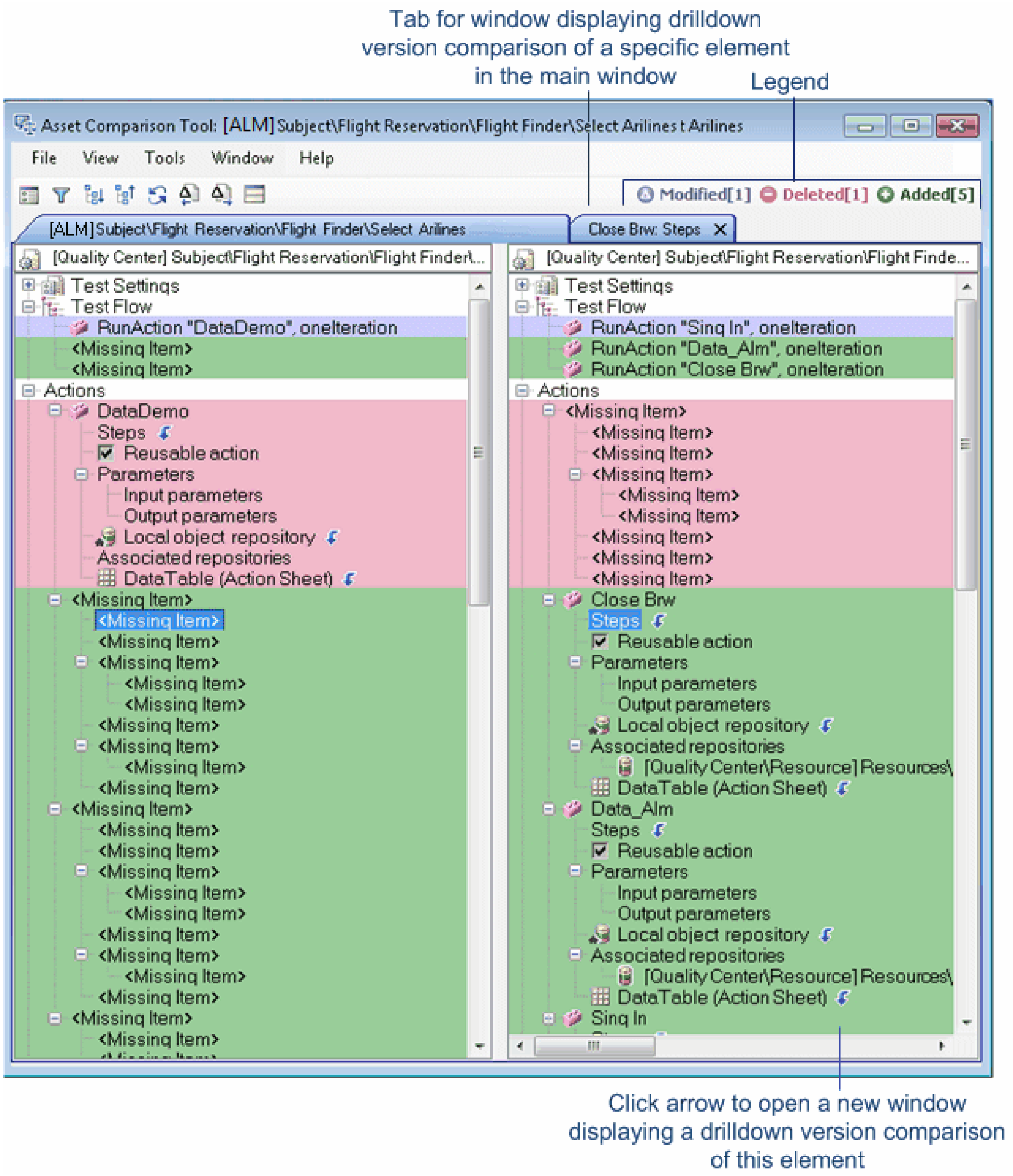
Reference

Asset Comparison Tool

Relevant for: GUI tests and components and API testing

This window enables you to compare two versions of a particular UFT asset, such as an application area, a test, a component, a function library, a shared object repository, a data table, a recovery scenario, or an environment variable XML file. It also enables you to drill down in an asset to view a comparison of entities that are associated with the asset, for example, an associated data table or shared object repository.

The following image shows an example of a test comparison in the Asset Comparison Tool. When viewing a comparison of a component, there are fewer items displayed. For example, a component comparison does not display an associated resources section and a test flow section.



Tab for window displaying drilldown version comparison of a specific element in the main window

Legend

Click arrow to open a new window displaying a drilldown version comparison of this element

To access	"How to Open the Asset Comparison Tool" on page 774
Important information	• To open the Asset Comparison Tool in UFT or ALM, the asset must be stored in an ALM project that has version control enabled. • The Asset Comparison Tool does not enable you to drill down to view assets that are associated via a relative path. For details, see "Relative Paths and ALM" on page 758.

Relevant tasks	"How to Work with the Asset Comparison Tool and Asset Viewer" on page 778
See also	• "Asset Viewer" on page 788 • "Color Settings Dialog Box (Asset Comparison Tool and File Content Checkpoint Preview)" on page 785 • "Filter Dialog Box (Asset Comparison Tool and File Content Checkpoint Preview)" on page 787

User interface elements are described below:

Menu, Toolbar, and Button Options

UI Elements		**Description**
	File > Exit	Closes the Asset Comparison Tool window. **Shortcut key:** ALT+F4
	View > Next Difference	Finds the next difference between the elements in the compared versions. **Shortcut key:** CTRL+DOWN ARROW
	View > Previous Difference	Finds the previous difference between the elements in the compared versions. **Shortcut key:** CTRL+UP ARROW
	View > Expand All or Collapse All	Expands or collapses the tree of asset properties.
	View > Refresh	Performs a new comparison of the selected asset versions. **Note:** This is useful if you are comparing the current version of an asset. If you modify and save the asset, you can use the **Refresh** command to view an updated comparison. **Shortcut key:** F5
	Tools > Color Settings	Opens the Color Settings dialog box (described on page 785), enabling you to define the text and background color for each filter type.

UI Elements		Description
▽	**Tools > Filter**	Opens the Filter dialog box (described on page 787), enabling you to show or hide the following types of filter elements in the comparison window: • **Modified** △ • **Deleted** ⊖ • **Added** ⊕ • **Identical** **Tip:** The "Legend" on the next page in the top-right corner of the comparison window indicates how many elements match each filter type. The legend adjacent to a collapsed node indicates how many sub-nodes match each filter type. For details, see Legend.
	Window > Close Tab	Closes the currently active comparison tab if it was opened from the main window. Enabled only if more than one tab is open. **Tip:** • You can open another tab to view a comparison of an asset that is associated with the currently compared asset, such as a shared object repository or data table. You do this by clicking the blue drilldown arrow ↓ adjacent to any asset that can be compared. • You can also close the comparison tab by clicking the **X** in the tab at the bottom of the window.
▤	**Window > View Horizontal or View Vertical**	**View Horizontal.** Displays the open documents one above the other. **View Vertical.** Displays the open documents side-by-side.
	Window > <Compared Asset Path>	Enables you to navigate between the open comparison tabs.
	Help > Asset Comparison Tool Help	Opens the Asset Comparison Tool Help. **Shortcut key:** F1

UI Elements		Description
	Previous 2000 Lines	If the testing document has more than 2000 lines, this button is displayed at the top of the comparison pane. Click to hide the current 2000 lines and display the previous 2000 lines of the testing document.
	Next 2000 Lines	If the testing document has more than 2000 lines, this button is displayed at the bottom of the comparison pane. Click to hide the current 2000 lines and display the next 2000 lines of the testing document.

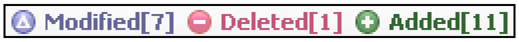
Shortcut Menu Commands (GUI testing only)

Command	Description
View Drilldown of Selected Asset	Opens a drilldown version comparison of the selected asset in a new tab. (Relevant only for assets that can be compared.) **Shortcut key:** ENTER **Tip:** • You can also click the blue drilldown arrow ↓ adjacent to the node to open a drilldown version comparison in a new tab. • You cannot drill down to view assets that are associated via a relative path. For details, see "Relative Paths and ALM" on page 758.
View Sample Snapshot	Opens a window containing a sample image of the selected element in UFT, for example, the Resources pane in the Test Settings or Business Component Settings dialog box. The element itself is highlighted in the snapshot. **Shortcut key:** CTRL+Q

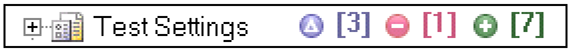
Legend

If the sub-elements of an element are different between versions, and you collapse the node representing that element, a legend is displayed adjacent to the node. This legend indicates the number of differences that exist under the collapsed element.

The following is an example of the filter legend displayed in the top-right corner of the Asset Comparison Tool window:



Modified[7] Deleted[1] Added[11]

Important information	• If you modify the background color of a filter type (using the Color Settings dialog box), the color of the filter type in the legend changes accordingly. • If you collapse an asset in the comparison window, the tool displays a legend for that asset, as shown in the following example:  ⊞ Test Settings Δ [3] ⊖ [1] ⊕ [7]
Relevant tasks	"How to Work with the Asset Comparison Tool and Asset Viewer" on page 778
See also	"Color Settings Dialog Box (Asset Comparison Tool and File Content Checkpoint Preview)" below

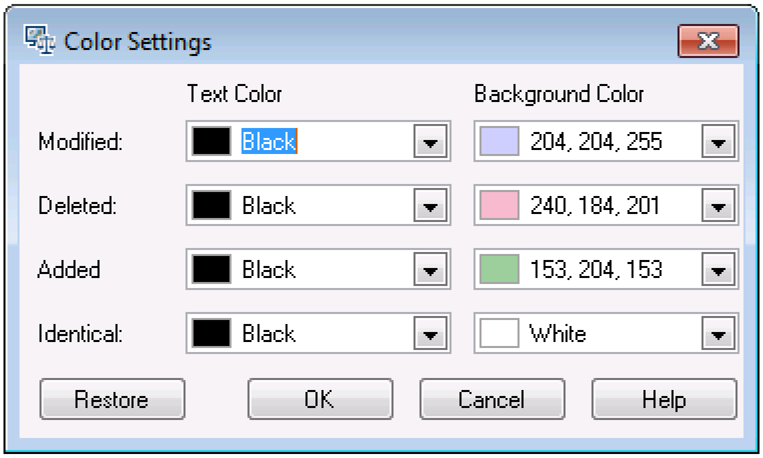
User interface elements are described below:

Symbol	Description	Number
Δ	**Modified**	Indicates the number of modified elements in the comparison.
⊖	**Deleted**	Indicates the total number of elements that were deleted from either of the versions being compared.
⊕	**Added**	Indicates the total number of elements that were added to either of the versions being compared.

Color Settings Dialog Box (Asset Comparison Tool and File Content Checkpoint Preview)

Relevant for: GUI tests and components and API testing

This dialog box enables you to modify the text and background colors for the various filter elements in the Asset Comparison Tool window and the File Content Checkpoint Preview (for GUI testing only). The changes remain in effect for all subsequent sessions.



Color Settings

	Text Color	Background Color
Modified:	Black	204, 204, 255
Deleted:	Black	240, 184, 201
Added	Black	153, 204, 153
Identical:	Black	White

To access	In the Asset Comparison Tool window: • Select the **Tools > Color Settings** menu command. • Click the **Color Settings** toolbar button . **For GUI testing:** In the File Content Checkpoint Preview, click the **Color Settings** toolbar button .
Important information	If you change the background color for a filter type, the legend in the top-right corner of the window changes accordingly. These changes remain in effect unless you change them again or restore the default settings.
See also	• "Asset Comparison Tool" on page 780 • "Checkpoint Properties Dialog Box" on page 1432

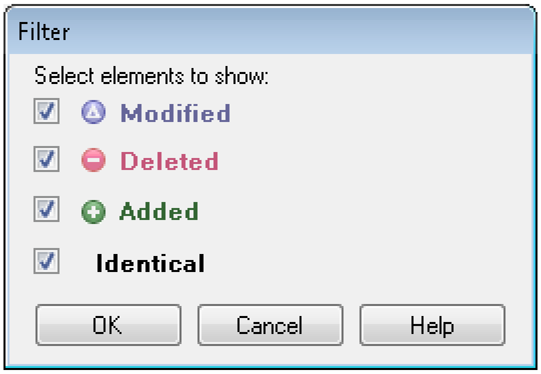
User interface elements are described below:

UI Elements	Description
Modified/ Deleted/ Added/ Identical	Text color and background color for the relevant filter elements. You can: • Click a down arrow ▼ to select a color from the list of colors in the Custom, Web, or System tabs. • Enter an RGB value directly in the edit box.
Restore	Restores the default color values for each of the filter elements.

Filter Dialog Box (Asset Comparison Tool and File Content Checkpoint Preview)

Relevant for: GUI tests and components and API testing

This dialog box enables you to show or hide elements in the comparison window according to filter criteria.



To access	In the Asset Comparison Tool window: - Select the **Tools > Filter** menu command. - Click the **Filter** toolbar button . In the File Content Checkpoint Preview, click the **Filter** toolbar button .
See also	- "Asset Comparison Tool" on page 780 - "Checkpoint Properties Dialog Box" on page 1432

User interface elements are described below:

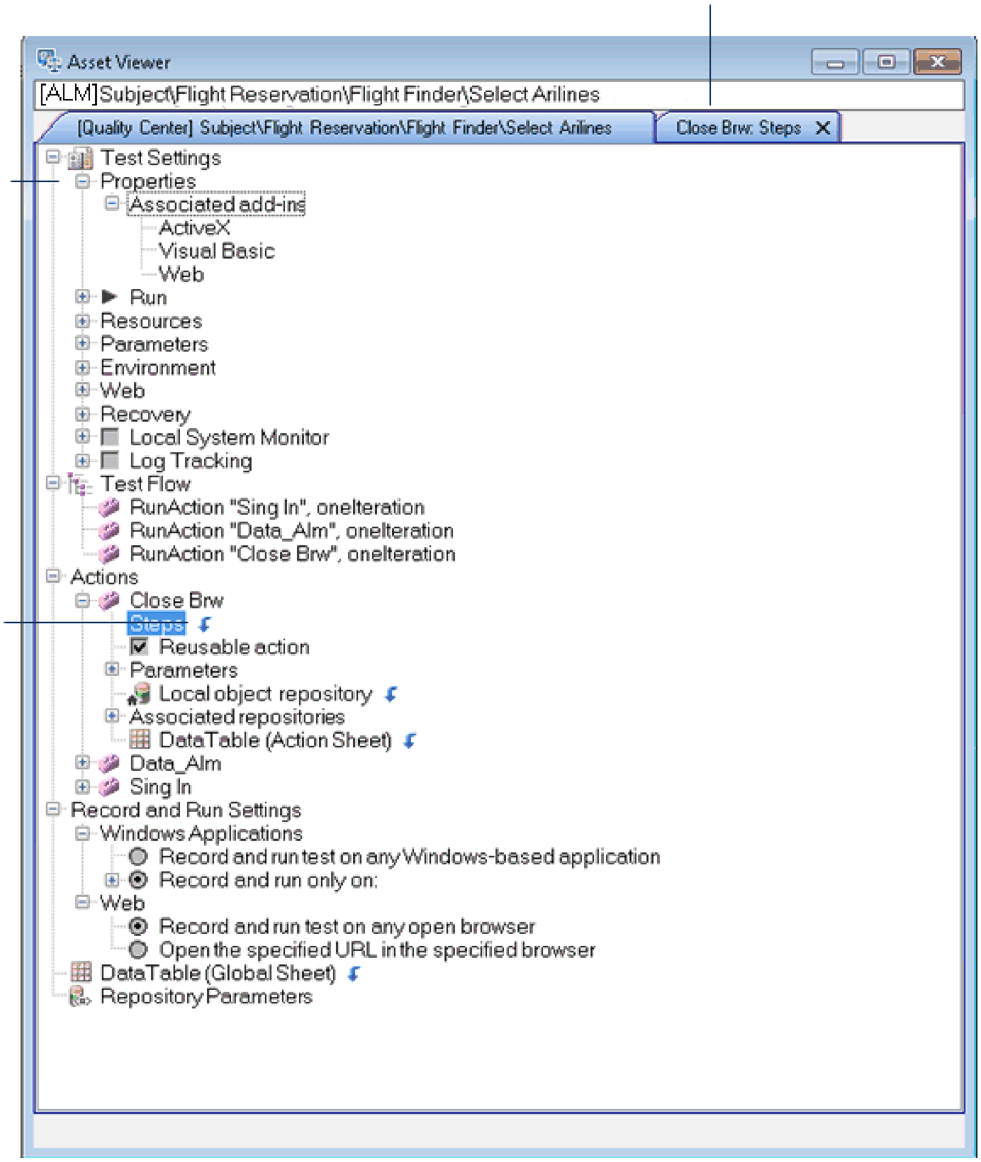
UI Elements	Description
Select elements to show	Select or clear a check box. The comparison window displays only those elements that match the defined filter. You can show or hide the following types of elements: - **Modified** - **Deleted** - **Added** - **Identical**

Asset Viewer

Relevant for: GUI tests and components and API testing

This window provides a functional overview of an asset, enabling you to view its configurations and settings in a viewer format. The tree view enables you to drill down to view or verify a particular setting without needing to open different dialog boxes or even UFT.

The image displayed below shows the Asset Viewer for a GUI test. When viewing a component in the Asset Viewer, only the following nodes are displayed: **Settings**, **Properties**, **Parameters**, **Local Object Repository**, **Repository Parameters**



Tab for window displaying drilldown version of a specific element in the main window

Expand a node to view its content

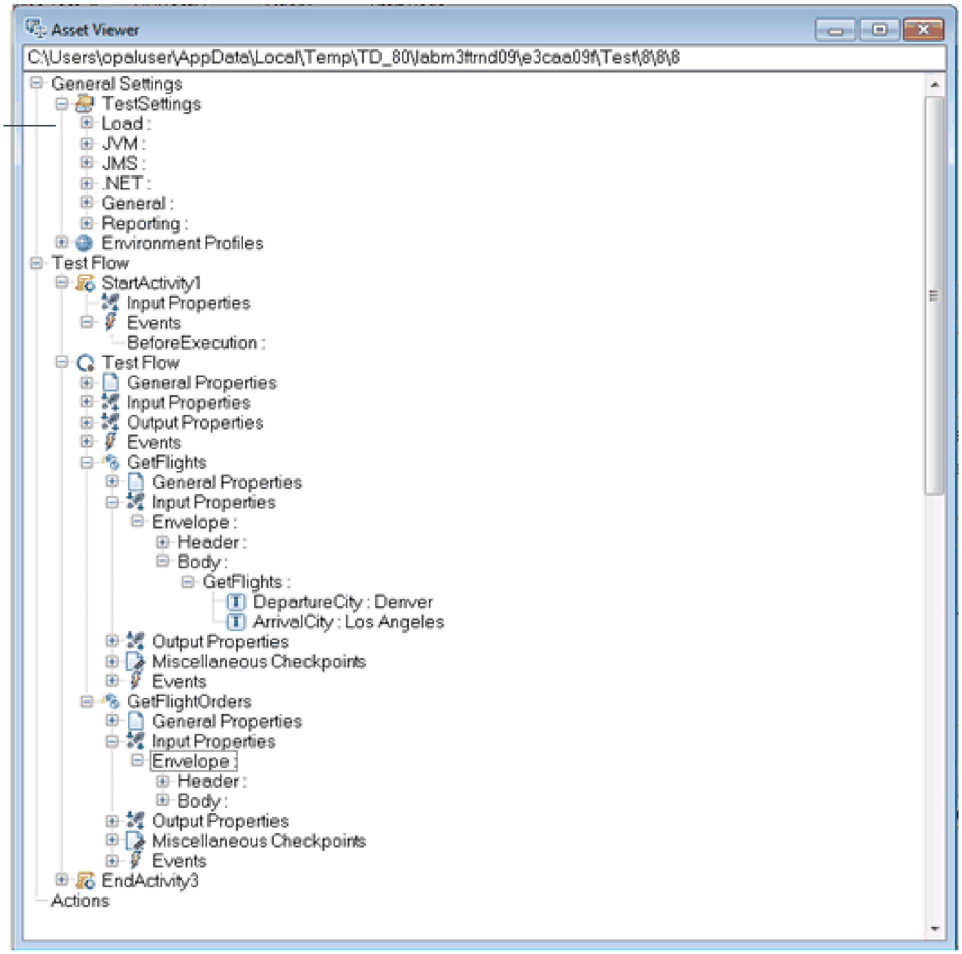
Click arrow to open a new window displaying a drilldown version of this element

[ALM]Subject\Flight Reservation\Flight Finder\Select Arilines

[Quality Center] Subject\Flight Reservation\Flight Finder\Select Arilines
Close Brw: Steps

- Test Settings
  - Properties
    - Associated add-ins
      - ActiveX
      - Visual Basic
      - Web
  - ▶ Run
  - Resources
  - Parameters
  - Environment
  - Web
  - Recovery
  - Local System Monitor
  - Log Tracking
- Test Flow
  - RunAction "Sing In", oneIteration
  - RunAction "Data_Alm", oneIteration
  - RunAction "Close Brw", oneIteration
- Actions
  - Close Brw
    - Steps
    - ☑ Reusable action
    - Parameters
    - Local object repository
    - Associated repositories
    - DataTable (Action Sheet)
  - Data_Alm
  - Sing In
- Record and Run Settings
  - Windows Applications
    - ○ Record and run test on any Windows-based application
    - ● Record and run only on:
  - Web
    - ● Record and run test on any open browser
    - ○ Open the specified URL in the specified browser
- DataTable (Global Sheet)
- Repository Parameters

The image below shows the Asset Comparison tool for an API test.



Expand a node to view its content

```
C:\Users\opaluser\AppData\Local\Temp\TD_80\labm3ftmd09\e3caa09f\Test\8\8
⊟ General Settings
    ⊟ TestSettings
        ⊞ Load :
        ⊞ JVM :
        ⊞ JMS :
        ⊞ .NET :
        ⊞ General :
        ⊞ Reporting :
    ⊞ Environment Profiles
⊟ Test Flow
    ⊟ StartActivity1
        Input Properties
        ⊟ Events
            BeforeExecution :
    ⊟ Test Flow
        ⊞ General Properties
        ⊞ Input Properties
        ⊞ Output Properties
        ⊞ Events
        ⊟ GetFlights
            ⊞ General Properties
            ⊟ Input Properties
                ⊟ Envelope :
                    ⊞ Header :
                    ⊟ Body :
                        ⊟ GetFlights :
                            DepartureCity : Denver
                            ArrivalCity : Los Angeles
            ⊞ Output Properties
            ⊞ Miscellaneous Checkpoints
            ⊞ Events
        ⊟ GetFlightOrders
            ⊞ General Properties
            ⊟ Input Properties
                ⊟ Envelope :
                    ⊞ Header :
                    ⊞ Body :
            ⊞ Output Properties
            ⊞ Miscellaneous Checkpoints
            ⊞ Events
    ⊞ EndActivity3
Actions
```

To access	"How to Open the Asset Viewer" on page 776
Important information	• To open the Asset Viewer in UFT or ALM, the asset must be stored in an ALM project that has version control enabled. • The Asset Viewer does not enable you to drill down to view assets that are associated via a relative path. For details, see "Relative Paths and ALM" on page 758.
Relevant tasks	"How to Work with the Asset Comparison Tool and Asset Viewer" on page 778
See also	"Asset Comparison Tool" on page 780

User interface elements are described below:

Button Options

UI Elements	Description
Previous 2000 Lines	If the testing document has more than 2000 lines, this button is displayed at the top of the pane. Click to hide the current 2000 lines and display the previous 2000 lines of the testing document.
Next 2000 Lines	If the testing document has more than 2000 lines, this button is displayed at the bottom of the pane. Click to hide the current 2000 lines and display the next 2000 lines of the testing document.

Context Menu Commands (GUI testing only)

Command	Description
View Drilldown of Selected Asset	Opens a drilldown version comparison of the selected asset in a new tab. (Relevant only for assets that can be compared.) **Shortcut key:** Enter **Tip:** • You can also click the blue drilldown arrow ↓ adjacent to the node to open a drilldown version comparison in a new tab. • You cannot drill down to view assets that are associated via a relative path. For details, see "Relative Paths and ALM" on page 758.
View Sample Snapshot	Opens a window containing a sample image of the selected element in UFT, for example, the Resources pane in the Test Settings or Business Component Settings dialog box. The element itself is highlighted in the snapshot. **Shortcut key:** Ctrl+Q

Troubleshooting and Limitations - Asset Comparison Tool

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for the Asset Comparison Tool and the Asset Viewer.

- **For GUI testing only:** Sometimes, checkpoint and output value comparison information disappears from the bottom panes of the Asset Comparison Tool after refreshing it. This can occur if you:

  a. Open the Asset Comparison Tool to view a comparison.

  b. Switch to the Object Repository Manager.

  c. Modify any of the current comparison's checkpoints or output values.

  d. Save your changes.

  e. Switch back to the Asset Comparison Tool.

  f. Refresh the Asset Comparison Tool.

  g. Select the checkpoint or output value that you modified.

  Workaround: If this occurs, close the Asset Comparison Tool, and then open it again after you save your changes.

- When comparing two baselines, if the only change in a resource is its association to a test or component, the Asset Comparison Tool does not indicate any change in the resource even though ALM may indicate that the resource is **Modified**.

- **For GUI testing only:** When working with a localized installation of UFT, selecting the **View Sample Snapshot** option in the UFT Asset Comparison Tool opens a window containing a sample image of the selected element in UFT. The image displays the English user interface.

- **For API testing:** The Asset Comparison tool is not available for API components.

- **For API testing:** The Asset Comparison tool does not show differences for the following API testing features:

  - Changes of properties for asynchronous Web service calls

  - Changes in multipart properties for HTTP Request steps

  - Changes in XML fault checkpoints

- Changes in attachments for Web service calls and SOAP Request steps
- Changes in security settings for Web services and SOAP Request steps

Chapter 28: Version Control in ALM

Relevant for: GUI tests and components, API testing, and business process tests and flows

> **Note:** The references to ALM in this chapter apply to Quality Center 10.00 and ALM. Note that some features and options may not be supported in the Quality Center or ALM edition you are using. For information on Quality Center or ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

Managing Versions of Assets in ALM Overview

Relevant for: GUI tests and components, API testing, and business process tests and flows

When UFT is connected to an ALM project with version control support, you can update and revise your UFT assets while maintaining earlier versions of each asset. This helps you keep track of the changes made to each asset and see what was modified from one version to another. Assets can include tests, components, function libraries, application areas, shared object repositories, recovery scenarios, and external data tables, XML files, user code files, or test activities.

You can check in the asset at any time. For example, you may want to check the asset in every day or when you complete a task. While the asset is checked out to you, other users can view the last checked in version of that asset in read-only mode, but they cannot modify the asset or view your changes until you check in the asset.

If the asset is...	You can...
checked in	• Open the asset in read-only mode using the **Open** option. You cannot modify the asset. • Open the asset and check it out by selecting **ALM > Check Out**. **Tip: (for GUI testing):** If a test, component, function library, or application area is checked into a project with version control, the document tab indicates its **(Read-Only)** status. **Note:** If you check out a test, and that test is associated with a data table that is currently checked in to the ALM project, the data in that data table is read-only, even though the test is editable. To modify the data table, you must first check out the data table in the ALM Test Resources module.
checked out to your ALM user name	Open the asset in read-write mode, using the **Open** option and modify the asset as needed. **Tip:** If a test, component, function library, or application area is checked out to your ALM user name, an unlocked icon 🔓 in the Solution Explorer indicates this status.

If the asset is...	You can...
checked out to another ALM user	Open the asset in read-only mode using the **Open** option. UFT displays a message indicating that the asset is checked out to another ALM user. You can view the last checked in version of the asset now, and you can check out the asset later after the other user checks in the asset. **Tip:** If a test, component, function library, or application area is checked out to another ALM user name, the document tab indicates this status by displaying a locked icon 🔒 and **(Read-Only)** adjacent to the document's name.

In UFT, you can check out only the latest version of an asset, although you can view and compare earlier versions. This is because assets that are stored in ALM are often linked to or **dependent on** one another.

For example, if you try to run an earlier version of a test or component with a later version of a shared object repository or a data table, your test or component might fail because the objects in the object repository or data table would not necessarily match the objects or steps in the test or component. For more details, see "Troubleshooting and Limitations - ALM Version Control" on page 807.

How ALM Manages Assets

Relevant for: GUI tests and components, API testing, and business process tests and flows

You manage asset versions by checking assets in and out of the version control database.

You add an asset to the version control database by saving it in an ALM project with version control support. When you save an asset for the first time, UFT automatically checks the asset into the ALM version control database, assigns it version number 1, and automatically checks the asset out for you so that you can continue working on it. When you check the asset in, the asset retains version number 1, since this is the first version that can contain content. Then, each time the asset is checked out and in again, the version number increases by 1.

> **Note:** If you create an asset directly in ALM, the asset is assigned version number 1 and is immediately checked out to you. In ALM, version number 1 represents the created asset without content. When you next check the asset in, ALM assigns it version number 2.

View and Compare Asset Versions

Relevant for: GUI tests and components and API testing

You can view and compare the versions of an asset using the Asset Comparison Tool. For details, see "Viewing and Comparing Versions of UFT Assets" on page 770.

If your project administrator creates project baseline versions when a milestone is reached during product development, you can view and compare the asset versions stored in these baselines. For details, see "Viewing Baseline History" on the next page.

> **Note:** With the exception of the **Baseline History** option, the **ALM Version Control** options in the **ALM** menu are available only when you are connected to an ALM project with version control support, and an asset stored in ALM is open in the UFT window.

Adding Assets to the Version Control Database in an ALM Project

Relevant for: GUI tests and components, API testing, and business process tests and flows

When you create a new asset or use **Save As** to save an existing asset in an ALM project with version control support, UFT automatically saves the asset in the project, checks the asset into the version control database with version number 1, and then checks it out so that you can continue working. This is an administrative version of the asset, similar to a placeholder. The version number indicates that the asset exists in the database. When you later check in the asset, the version number remains version number 1—the first version that you are checking in. Subsequent checkins increase the version number by 1.

Saving your changes to an existing asset does not check them in. Even if you save and close the asset, the asset remains checked out until you choose to check it in. For details, see "Checking Assets into the Version Control Database" on the next page.

Checking Assets Out of the Version Control Database

Relevant for: GUI tests and components, API testing, and business process tests and flows

When you open an asset that is currently checked in to the version control database, it is opened in read-only mode. You can review the checked-in asset. You can also run the asset and view the results.

To modify the asset, you must check it out. When you check out an asset, ALM copies the asset to your unique check-out folder (automatically created the first time you check out an asset), and locks the asset in the project database. This prevents other users of the ALM project from overwriting any changes you make to the asset. However, other users can still run the version that was last checked in to the database.

You can save and close the asset, but it remains locked until you return the asset to the ALM database. To release the asset, either check the asset in, or undo the check out operation. For more details on checking assets in, see "Checking Assets into the Version Control Database" on the next page. For details on undoing the check-out, see "How to Cancel a Check-Out Operation" on page 801.

In UFT, the check out option accesses the latest version of the asset. In ALM, you can also check out earlier versions of any asset except for application areas. For details, see "Version History Dialog Box" on page 803 and the ALM user guide.

Checking Assets into the Version Control Database

Relevant for: GUI tests and components, API testing, and business process tests and flows

While an asset is checked out, ALM users can run the previously checked-in version of your asset. For example, suppose you check out version 3 of an asset and make a number of changes to it and save the asset. Until you check the asset back into the version control database as version 4, ALM users can continue to run version 3.

When you have finished making changes to an asset and you are ready for ALM users to use your new version, you check it in to the version control database.

> **Note:** If you do not want to check your changes into the ALM database, you can undo the check-out operation.

When you check an asset back into the version control database, ALM deletes the asset copy from your checkout folder and unlocks the asset in the database so that the asset version is available to other users of the ALM project.

Viewing Version Control Information When Opening a Test

Relevant for: GUI tests and components, API testing, and business process tests and flows

When you open a test from an ALM project with version control support, you can view version control information for the test by using the **Details** view in the "Open/New <Document>/<Resource> Dialog Box" (described on page 156).

The **Checked Out To** column specifies the user name of the ALM user to whom the test is checked out, if it is checked out. If the test is currently checked in to the version control database, there is no indication in the dialog box.

Viewing Baseline History

Relevant for: GUI tests and components and API testing

In ALM, a project administrator can create baselines that provide "snapshots" of an entire project (or part of a project) at different stages of development. A **baseline** represents a version of a project at a specific point in a project's life cycle. For example, baselines are often created for each milestone or when specific phases in a project are completed.

Baselines can be created for ALM projects that are enabled for version control, and for projects for which version control is not enabled.

The project administrator creates the baseline in the Libraries tab of the Management module in ALM. Creating a baseline is a two-fold process. The administrator first creates a library, which specifies the root folders from which to import the data. The administrator makes sure to include all of the associated resource files, such as shared object repositories and function libraries. The administrator then creates the actual baseline, which comprises the latest versions of every asset included in the library. If the project is version control-enabled, then these are the latest checked in versions of every asset.

During the creation process, ALM verifies that all of these assets (such as associated resource files) are included in the baseline. If any assets are not included, ALM informs the administrator so that the library and baseline can be modified accordingly. For more details, see the ALM user guide.

In ALM, these baselines can be viewed and compared in their entirety.

In UFT, you can view and compare the assets saved in these baselines. This enables you to review the content of an asset at a specific phase in the project time line.

You can also run a test or component from a baseline.

Version History Versus Baseline History

Relevant for: GUI tests and components and API testing

This section focuses on the differences between version history and baseline history and describes when to use each.

- You use version control to check in and check out assets as needed. For example, you may want to check in an asset on a daily basis or only when significant results are achieved. This enables you to monitor the asset's development.

  If you want to view the content of an asset on a particular date or after a particular user checked in the asset, use the **Version History** option to view or compare the asset.

- The ALM project administrator creates baselines that represent "snapshots" of a project's assets at various milestones in a project's life cycle. Each baseline links to the assets specified by the administrator when the baseline was created. The asset version represented in the baseline is always the version that was checked in when the baseline was created.

  If you want to view an asset as it was saved for a particular milestone, use the **Baseline History** option.

Tasks

How to Manage Version Control Operations

Relevant for: GUI tests and components, API testing, and business process tests and flows

This task describes how to check in the currently open asset, check out the latest version of an asset in order to edit it, and cancel a check-out operation if needed.

After you edit an asset, you can save changes and close it without checking in the modified asset. However, your changes are not available to other ALM users until you check it in. If you do not want to check your changes in, you can undo the check-out by canceling it.

For more details on checking assets in, see "Checking Assets into the Version Control Database" on page 798.

This task includes the following steps:

- "How to Check In the Currently Open Asset" below
- "How to Check Out the Latest Version of an Asset" below
- "How to Cancel a Check-Out Operation" on the next page

How to Check In the Currently Open Asset

1. Confirm that the currently open asset is checked out to you. For details, see "Version History Dialog Box" on page 803.

   > **Note:** If the open asset is currently checked in, the **Check In** option is disabled. If you open an asset that is checked out to another user, all **ALM Version Control** options, except the **Version History** option, are disabled.

2. Select **ALM > Check In**, and check in the asset using the Check In dialog box. For details, see "Check In/Check Out Dialog Box" on page 802.

How to Check Out the Latest Version of an Asset

1. Make sure the asset you want to check out is currently checked in. If you open an asset that is checked out to you, the **Check Out** option is disabled. If you open an asset that is checked out to another user, all ALM version control options, except the **Version History** option, are disabled.

   > **Note: Note about version numbers:** Prior to Quality Center 10.00, version numbers consisted of three segments separated by periods, for example 1.7.4. In

> Quality Center 10.00 and ALM, version numbers consist of a single segment, for example 12.

2. Open the "Open/New <Document>/<Resource> Dialog Box" as follows:

If the asset is a:	Do this:
Test, Component or Function Library	In the main UFT window, select **File > Open >** and select **Test**, **Business Component**, **Function Library**, or **Application Area** or click the **Open** down arrow and select the asset type from the list.
Shared Object Repository (**GUI testing only)**	In the Object Repository Manager, select **File > Open** or click the **Open** button.
Recovery Scenario (**GUI testing only)**	In the Recovery Scenario Manager, click the **Open** button.

3. Browse to and open the asset.

   The document opens in the document pane as read-only with a lock icon in the document's tab.

4. Select ALM **> Check Out** to check out the document and edit it.

How to Cancel a Check-Out Operation

1. Open the asset if it is not already open.

2. Select **ALM > Undo Check out.**

3. Click **Yes** to confirm the cancellation of your check out operation.

   The check out operation is cancelled. The checked out asset closes, and the previously checked in version reopens in read-only mode.

Reference

Version Management Commands

Relevant for: GUI tests and components, API testing, and business process tests and flows

The following version control commands are available in UFT and can be used when connected to an ALM project that has version control enabled:

- **Check Out.** Enables you to check a version-controlled asset out of the version control database. For details, see "Checking Assets Out of the Version Control Database" on page 797.

- **Undo Check Out.** Enables you to cancel the check out of a version-controlled asset from the version control database.

- **Check In.** Enables you to check an asset in to the version control database. For details, see "Checking Assets Out of the Version Control Database" on page 797.

- **Version History.** Enables you to view or compare the versions of a particular asset. For details, see "Managing Versions of Assets in ALM Overview" on page 795.

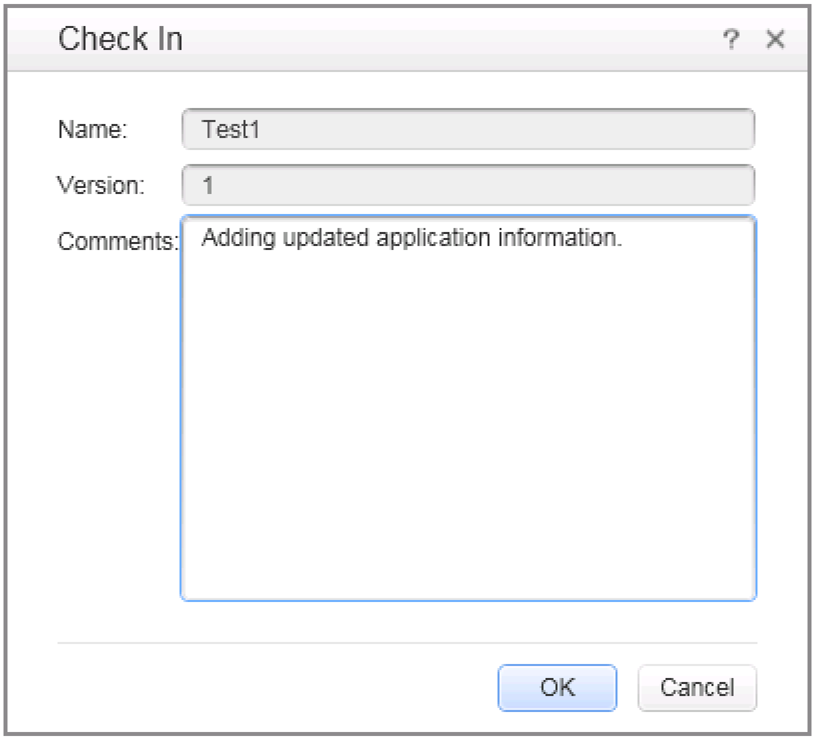
- **Baseline History. (GUI tests and components, API testing only)** Enables you to view or compare the versions of a particular asset as it was saved in a project's baselines. For details, see "Viewing Baseline History" on page 798.

Check In/Check Out Dialog Box

Relevant for: GUI tests and components, API testing, and business process tests and flows

This dialog box enables you to check in or check out an ALM asset in to or from the ALM version control database.

The image below shows an example of the Check-in dialog box for a version-controlled test.



Check In

Name: Test1

Version: 1

Comments: Adding updated application information.

To access	ALM > Check In or Check Out
Relevant tasks	- "How to Check In the Currently Open Asset" on page 800 - "How to Check Out the Latest Version of an Asset" on page 800

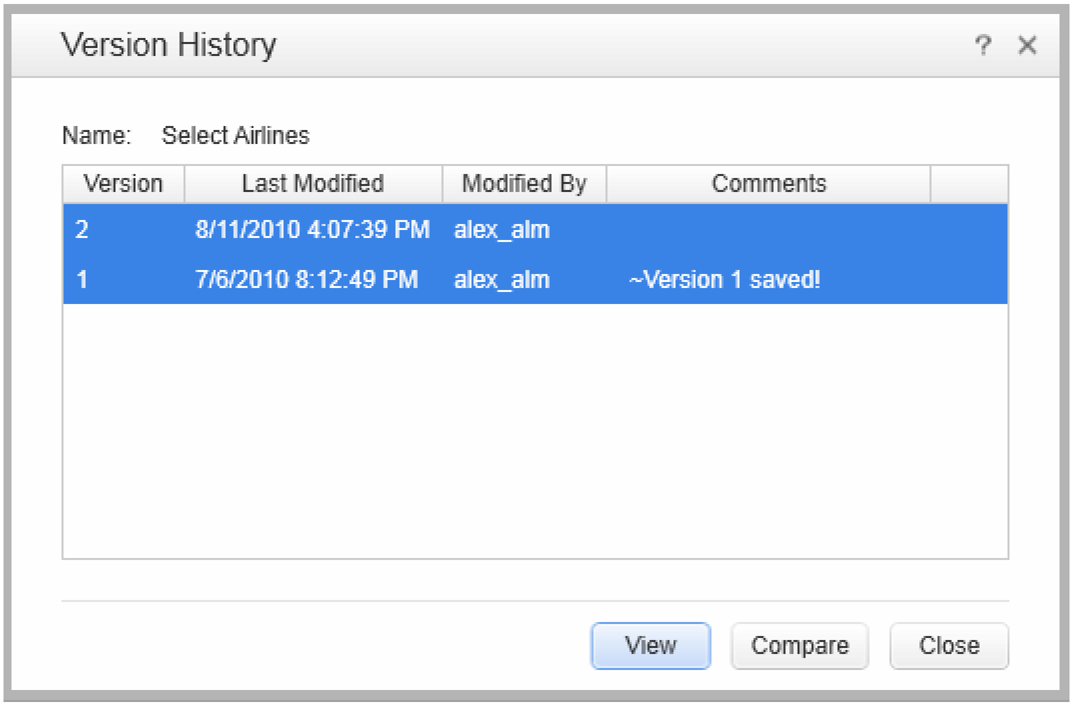
User interface elements are described below:

UI Elements	Description
Name	The asset name.
Version	The new version number. By default, the new version number is one number higher than the previously checked in version.
Comments	If you entered a description of your change when you checked out the asset, the description is displayed in the **Comments** box. You can enter or modify the comments in the box.

Version History Dialog Box

Relevant for: GUI tests and components, API testing, and business process tests and flows

This dialog box enables you to view the version history for an asset, and depending on the type of asset, view the content of a previous asset version and compare two asset versions.



Name: Select Airlines

Version	Last Modified	Modified By	Comments
2	8/11/2010 4:07:39 PM	alex_alm	
1	7/6/2010 8:12:49 PM	alex_alm	~Version 1 saved!

To access	- **From most assets:** Open the asset and select the **ALM > Version History** menu command. - **From a recovery scenario (GUI testing only):** In the Recovery Scenario Manager, open the recovery scenario, click the **Version Control** down arrow, and select **Version History.**
Important information	**To view a version for an asset:** Select a version and click **View**. **To compare two versions of an asset:** Select two versions and click **Compare**. You cannot check out an earlier version of an asset from this dialog box. (You can check out earlier version of most assets directly from the ALM project. For details on checking out assets from ALM, see the ALM user guide.)
See also	"Managing Versions of Assets in ALM Overview" on page 795

User interface elements are described below:

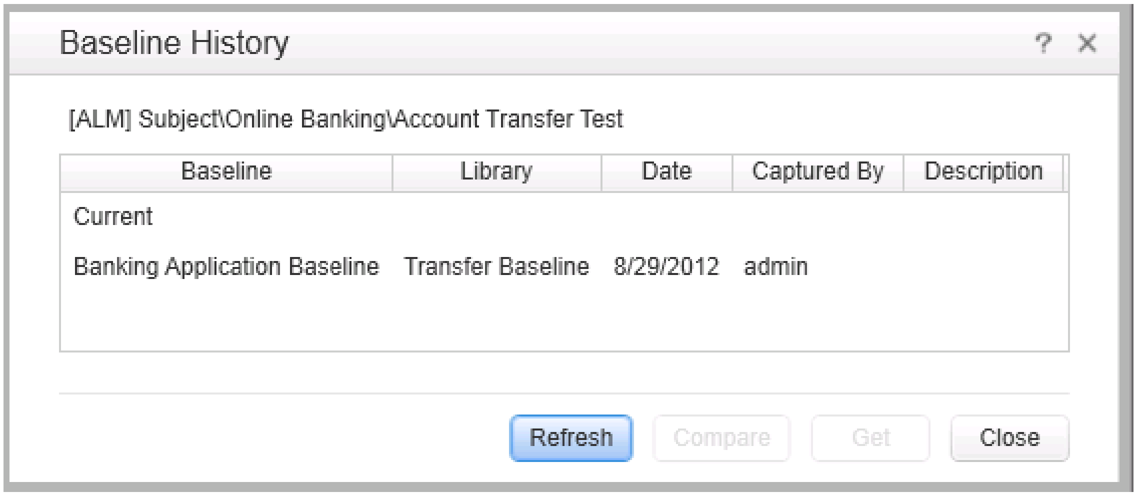
UI Elements	Description
Name	The name of the currently open asset.
Version column	A list of all versions of the asset.
Last Modified column	The date that each version was checked in.
Modified By column	The user who checked in each listed version.

UI Elements	Description
Comments column	The comments that were entered when the selected asset version was checked in.
View button (GUI tests and components and API testing only)	Enables you to view the selected version of the current asset. To view a version of an asset: Select an asset version and click View. UFT opens the checked in version of the asset in the Asset Viewer. For details on the Asset Viewer, see "Asset Viewer" on page 788.
Compare button (GUI tests and components and API testing only)	Enables you to compare two versions of the currently open asset. **To compare two versions:** Select the versions you want to compare and click **Compare**. UFT opens the two asset versions in the "Asset Comparison Tool" on page 780.

Baseline History Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to view and compare read-only baseline "snapshots" of an asset.



[ALM] Subject\Online Banking\Account Transfer Test

Baseline	Library	Date	Captured By	Description
Current				
Banking Application Baseline	Transfer Baseline	8/29/2012	admin	

To access	- **Most assets:** Open the asset and select the **ALM > Baseline History** menu command. - **Recovery scenario (GUI testing only):** In the Recovery Scenario Manager, open the recovery scenario, click the **Version Control** down arrow, and select **Baseline History.**
Important information	In the ALM Test Lab module, you can use the **Pin to Baseline** option to run a baseline version of an asset. For more details, see the ALM user guide.
See also	"Viewing Baseline History" on page 798

User interface elements are described below:

UI Elements	Description
Refresh button	Reloads the baselines in the Baseline History dialog box with the latest changes. For example, if a baseline is added while this dialog box is open, clicking **Refresh** updates the list of baselines.
Baseline column	Lists all of the baselines that include this asset. Baselines are defined in the ALM project (**Management** module > **Libraries** tab).
Library column	Lists the libraries from which each baseline was created.
Date column	Lists the date that each baseline was created.
Captured By column	Lists the ALM user who created each listed baseline.
Description column	Displays any comments that were added when the baseline was created.
Compare button	Enables you to view a comparison of the currently open asset in two baselines. **To compare two baselines:** Select the baselines you want to compare and click **Compare**. UFT opens the two baseline versions of the asset in the "Asset Comparison Tool" on page 780.
Get button	Enables you to open the current asset from the selected baseline. **To view the asset as it was stored in a baseline:** Select a baseline from the list and click **View**. When you click **Get**, UFT: • Closes the currently open asset. • Opens the same asset from the baseline you selected. • Loads the baseline version of the external actions and resource files that are associated with the asset, if any, when they are called. **Note:** If an external GUI action or resource file is associated via a relative path, loads the latest version of the action or resource file instead of the version from the baseline.

Troubleshooting and Limitations - ALM Version Control

Relevant for: GUI tests and components, API testing, and business process tests and flows

This section describes troubleshooting and limitations for ALM version control.

- If you need to check out an earlier version of an asset, for example, to roll back to an earlier version, contact your ALM project administrator. Your administrator needs to ensure that the correct versions of all relevant assets become the latest versions.

- When working with a version-control-enabled ALM project, it takes a long time to save a test for the first time (up to twice as long as saving the same test in a project without version control support enabled). This delay does not occur on subsequent saves of the test.

- To enable API test versioning in Quality Center version 10.00, create a file in the <QC installation folder>\repository\sa\DomsInfo\Metadata\TEST folder called ServiceTest.xml with the following content:

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <Test Type="SERVICE-TEST">
    <Repository>
      <Folders ID="1" Filter="." BlindCopy="true" />
    </Repository>
    <Versioning Enabled="true" />
    <Baselining Enabled="false" />
  </Test>
  ```

Chapter 29: HP Sprinter

Relevant for: GUI tests only

This chapter includes:

Concepts

HP Sprinter Overview

Relevant for: GUI tests only

Although UFT automated tests can answer many of your testing needs, you may also need to perform some of your tests manually. You can run your manual tests using HP Sprinter, HP's solution for manual testing. Sprinter provides advanced functionality and tools to make manual testing more efficient and effective.

Manual testing often requires that you leave your testing application to accomplish tasks related to your test. For example, you may need to use graphic software to take a screen capture of your application, you may want to record a movie of the application during the test, and you need to switch to your defect tracking software to report defects.

Sprinter addresses these needs in the manual testing process, and enables you to accomplish these tasks without disrupting your test flow. Sprinter includes many tools to help you detect and submit defects. Sprinter can also perform many of the repetitive and tedious tasks of manual testing for you. These features ensure that you can perform all the tasks necessary for your manual test with minimum interruptions to your testing work.

Sprinter is fully integrated with ALM 11.00 and later, enabling you to get the maximum benefit from both solutions.

> **Note:** Unified Functional Testing (UFT) and Sprinter share a variety of system resources. Consider the following when deciding whether to install Sprinter on your UFT computer:
>
> - Sprinter and UFT can be installed on the same computer.
> - Sprinter and UFT cannot be run simultaneously on the same computer.
> - Any changes to the installation of one of these products will affect the other. If you uninstall, modify, or upgrade one product, the other may fail. You need to repair the installation of the affected product. For details, see the *HP Unified Functional Testing Installation Guide* and the *HP Sprinter Readme*.

With Sprinter you can:

- **Run ALM manual tests and Business Process tests with new step display:**
  - **Clear steps display.** Steps are presented in a clear, organized, and user-friendly design, making it easier to view step information and update step status.
  - **Move easily between tests in your run.** You can easily move between the tests in your run without interrupting your test flow. Sprinter updates all your displayed step and run information to match your current test.

    - **Edit actual parameter values during your test run.** You can easily edit the actual values of parameters in your test, during your test run.

    - **Multiple views.** Change the way you view your steps depending on your testing needs. View in normal mode when more details are needed, or view in Subtitles mode if you need to see more of your application.

    - **Actual value including screen captures.** Attach a plain or annotated screen capture of your application to the step's actual value.

- **Create formal tests from exploratory tests with no pre-defined steps.** If you run a test without pre-defined steps, Sprinter can keep a record of all the user actions you took during your test. You can then export this list to an Excel spreadsheet, modify the text as needed and import the spreadsheet to a test in ALM. This converts an exploratory test to a formal test, with pre-defined steps.

- **Submit defects to ALM.** Submit an ALM defect directly from within Sprinter. You can optionally let Sprinter create a defect scenario by automatically generating a text description of all the user actions or steps in your test. You can also attach a screen capture or a movie of your application to the defect.

- **Create and annotate screen captures of your application.** Sprinter provides tools that enable you to take and annotate a screen capture of your application at any point in the testing process. Tools are included that make measuring and comparing user interface elements easier. Report defects in the display by attaching the annotated screen capture to an ALM defect, saving it as a file, or attaching it to an email. You can also include annotated screen captures in the Actual Result of a step.

- **Let Sprinter perform some manual testing tasks for you.**

  You can create and run **macros** to automate a set of actions in your application. Sprinter can also **inject data** automatically into fields in your application.

- **Replicate your actions on another computer.** Replicate your user actions on multiple computers with different configurations (operating system, browser). Sprinter detects differences in the displays of these computers and enables you to report defects on these differences.

- **View test results.** Sprinter includes a powerful Storyboard that displays each action you performed in your test. For each action you can see a screen capture of the action, any defects that you reported, defect reminders, and comments. If you ran the test with multiple configurations you can view the differences between the displays of different computers.

For more details, contact your HP representative.

Part 6: GUI Testing Design

Chapter 30: GUI Test Creation Overview

Relevant for: GUI tests and components

This chapter includes:

Concepts

Methodologies for Creating Tests

Relevant for: GUI tests only

You can create tests using the keyword-driven methodology, step recording, importing steps from HP Sprinter, or a combination of all of these methods.

- The keyword-driven methodology enables you to select keywords to indicate the operations you want to perform on your application.
- Step recording enables you to record the operations you perform on your application.
- Importing steps from Sprinter enables you to convert manual tests to UFT GUI tests.

This section includes:

Creating Tests Using the Keyword-Driven Methodology

Relevant for: GUI tests only

This methodology requires an infrastructure for all of the required resources. Resources include shared object repositories, function libraries, and recovery scenarios. Setting up the infrastructure requires in-depth knowledge of your application and a high level of UFT expertise.

Although setting up the infrastructure may initially require a longer time-investment in comparison to recording tests, using the keyword-driven methodology enables you to create tests at a more application-specific level and with a more structured design. This enables you to maintain your tests more efficiently and provides you with more flexibility than a recorded test.

When to Use Keyword-Driven Testing

- Keyword-driven testing enables you to design your tests at a business level rather than at the object level. For example, UFT may recognize a single option selection in your application as several steps: a click on a button object, a mouse operation on a list object, and then a keyboard operation on a list sub-item. You can create an appropriately-named function to represent all of these lower-level operations in a single, business-level keyword.
- By incorporating technical operations, such as a synchronization statement that waits for client-server communications to finish, into higher level keywords, tests are easier to read and easier for less technical application testers to maintain when the application changes.

- Keyword-driven testing naturally leads to a more efficient separation between resource maintenance and test maintenance. This enables the automation experts to focus on maintaining objects and functions while application testers focus on maintaining the test structure and design.

- When you record tests, you may not notice that new objects are being added to the local object repository. This may result in many testers maintaining local object repositories with copies of the same objects. When using a keyword-driven methodology, you select the objects for your steps from the existing object repository. When you need a new object, you can add it to your local object repository temporarily, but you are also aware that you need to add it to the shared object repository for future use.

- When you record a test, UFT enters the correct objects, methods, and argument values for you. Therefore, it is possible to create a test with little preparation or planning. Although this makes it easier to create tests quickly, such tests are harder to maintain when the application changes and often require re-recording large parts of the test.

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

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

For details on creating tests using the keyword-driven methodology, see "Test Creation - Keyword-Driven Methodology" on page 820.

Creating Tests By Recording Steps on Your Application

Relevant for: GUI tests only

In some cases, you may want to let UFT generate test steps by recording the typical processes that you perform on your application.

As you navigate through your application, each step you perform is graphically displayed as a row in the Keyword View. A step is anything a user does that changes the content of a page or object in your application, for example, clicking a link or typing data into an edit box. Recording may be easier for new UFT users or when beginning to design tests for a new application or a new feature.

When to Record Tests

Recording can be useful in the following situations:

- Recording helps novice UFT users learn how UFT interprets the operations you perform on your application, and how it converts them to UFT objects and built-in operations.

- Recording can be useful for more advanced UFT users when working with a new application or major new features of an existing application (for the same reasons described above). Recording is also helpful while developing functions that incorporate built-in UFT keywords.

- Recording can be useful when you need to quickly create a test that tests the basic functionality of an application or feature, but does not require long-term maintenance.

For details on recording tests, see "Recording GUI Tests and Components - Overview" on page 843.

Creating Tests by Importing Steps from HP Sprinter

Relevant for: GUI tests only

Sprinter, HP's manual testing solution, enables the manual tester to perform operations (user actions) on an application while Sprinter captures and saves information about each user action and relevant test object in the background during the run session. This process is similar to recording steps on your application in UFT.

After the Sprinter run session ends, the manual tester can export the captured user actions, test objects, and comments, to an automated test data file in XML format.

You can then import this file to UFT, which converts it to a UFT GUI test with a local object repository. Each step in the new text represents a user action that was performed in Sprinter, and any comments that were added in Sprinter are displayed in the Comment column for the corresponding step in the Keyword View.

This method helps to increase testing coverage for the application, as it creates a more seamless workflow between manual testers and automation experts that are testing the same application.

For task details, see "How to Create and Manage Documents" on page 136.

Enhancing Your Tests

Relevant for: GUI tests only

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

You can also use a variety of options to enhance your existing tests. This section describes some of the ways in which you can enhance your existing tests.

Checkpoints

You can add checkpoints to your test. A **checkpoint** is a step in your test that compares specified items during a run session with the values stored for the same items within the test. This enables you to identify whether or not your application is functioning correctly. There are several different checkpoint types. For more details on creating checkpoints, see the **Checkpoints Overview** section in the *HP Unified Functional Testing User Guide*.

> **Tip:** You can also use the **CheckProperty** method, which enables you to verify the property

> value of an object without using the checkpoint interface. For details, see *HP UFT Object Model Reference for GUI Testing*.

Parameterization

When you test your application, you may want to check how it performs the same operations with different data. You can do this by replacing fixed values with values from an external source during your run session. This is called **parameterizing** your test. You can supply data from a data table, environment variables you define, or values that UFT generates during the run session. For more details, see "Parameterizing Object Values" on page 1526.

Output Values

You can retrieve values from your test and store them in the data table as output values. You can subsequently use these values as an input parameter in your test. This enables you to use data retrieved during a test in other parts of the test. For more details, see "Output Values Overview" on page 1488.

Actions

You can divide your test into actions to streamline the testing process of your application. For details, see "Actions in GUI Testing - Overview" on page 872.

Programming Statements

You can use special UFT options to enhance your test with programming statements. The Step Generator guides you step-by-step through the process of adding recordable and non-recordable operations (methods and properties) to your test. You can also synchronize your test to ensure that your application is ready for UFT to perform the next step in your test, and you can measure the amount of time it takes for your application to perform steps in a test by defining and measuring transactions. For more details, see "Generated Programming Operations" on page 968.

You can also manually enter standard VBScript statements, as well as statements using UFT test objects and operations, in the Editor. For details, see "Programming in GUI Testing Documents in the Editor" on page 994.

Active Screen Updates

As the content of your application changes, you can update the selected Active Screen display and use the Active Screen to add new steps to your test instead of re-recording steps on new or modified objects. For details, see "How to Update Test Object Descriptions, Checkpoints, or Output Values, or Active Screen Captures" on page 1088.

Editor and Keyword View - A Comparison

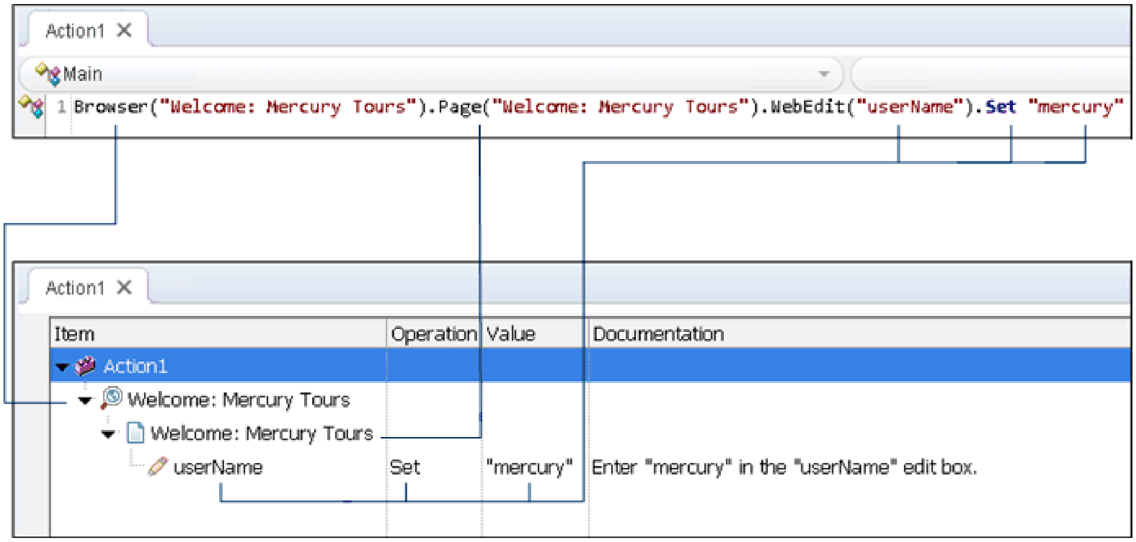
Relevant for: GUI actions and scripted GUI components

If you prefer to work with VBScript statements when editing actions or scripted components, you can choose to work in the Editor, as an alternative to using the Keyword View. You can move between the two views as you wish, by selecting the Editor / Keyword View toggle button.

The Editor displays the same steps and objects as the Keyword View, but in a different format:

- In the Keyword View, UFT displays information about each step and shows the object hierarchy in an icon-based table. For details, see "Keyword View" on page 919.
- In the Editor, UFT displays each step as a VBScript line or statement. In object-based steps, the VBScript statement defines the object hierarchy.

The following diagram shows how the same object hierarchy is displayed in the Editor and in the Keyword View:

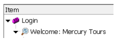
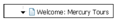
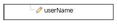
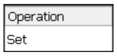
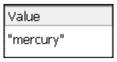


```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "mercury"
```

Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.

Each line of VBScript in the Editor represents a step. The example above represents a step in an action in which the user inserts the name mercury into an edit box. The hierarchy of the step enables you to see the name of the site, the name of the page, the type and name of the object in the page, and the name of the operation performed on the object.

The table below explains how the different parts of the same step are represented in the Keyword View and the Editor:

Keyword View	Editor	Explanation
 Item Login Welcome: Mercury Tours	Browser ("Welcome: Mercury Tours")	The name of the browser test object is Welcome: Mercury Tours.
 Welcome: Mercury Tours	Page ("Welcome: Mercury Tours")	The name of the current page is Welcome: Mercury Tours.
 userName	WebEdit ("userName")	The object type is WebEdit; the name of the edit box on which the operation is performed is userName.
 Operation Set	Set	The method performed on the edit box is **Set**.
 Value "mercury"	"mercury"	The value inserted into the **username** edit box is mercury.

For more details, see:

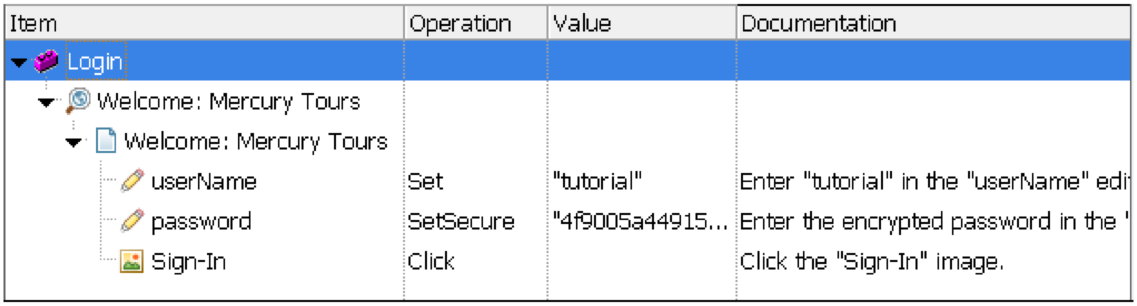
- "Checkpoint and Output Statements" on page 1013
- "Parameter Indications in VBScript" on page 1018
- "Basic VBScript Syntax" on page 1013
- "Automatic Code Completion" on page 316

Sample Test

Relevant for: GUI tests only

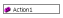
The following is a sample action of a login procedure to the Mercury Tours site, the sample Web site. When you create tests, UFT creates both a graphical representation and script of the steps you perform on your application.

The graphical representation of these steps is displayed in the Keyword View.



Item	Operation	Value	Documentation
Login			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"tutorial"	Enter "tutorial" in the "userName" edi
password	SetSecure	"4f9005a44915...	Enter the encrypted password in the '
Sign-In	Click		Click the "Sign-In" image.

The table below provides an explanation of each step in the Keyword View.

Step	Description
	**Action1** is the action name.
	The browser invokes the **Welcome: Mercury Tours** Web site.
	Welcome: Mercury Tours is the name of the Web page test object.
	userName is the name of the edit box test object. **Set** is the method performed on the edit box. **tutorial** is the **value** property of this edit box.
	password is the name of the edit box test object. **SetSecure** is an encryption method performed on the edit box. **4ff405198a68b24867227da98b21e547cfc0c2f47d31** is the encrypted value of the password.
	Sign-In is the name of the image link test object. Click is the method performed on the image. **26, 4** are the x- and y-coordinates where the image was clicked.

The Editor displays these same steps using a VBScript program based on the UFT object model.

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set
"tutorial"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").Set
Secure "4ff405198a68b24867227da98b21e547cfc0c2f47d31"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click 26,
4
```

Chapter 31: Test Creation - Keyword-Driven Methodology

Relevant for: GUI tests only

This chapter includes:

Concepts

Keyword-Driven Methodology - Overview

Relevant for: GUI tests only

Keyword-driven testing is a technique that separates much of the programming work from the actual test steps so that the test steps can be developed earlier and can often be maintained with only minor updates, even when the application or testing needs change significantly. This enables you to create structured tests that are easier to update and maintain over time.

The keyword-driven methodology is especially useful for organizations that have both technical and less technical users because it offers a clear division of automation tasks. This enables a few experts to maintain the resource framework while less technical users design and maintain automated test steps. Additionally, after the basic infrastructure is in place, both types of users can often do their jobs simultaneously.

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

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

The information in the sections listed below provides guidance on the main steps involved in creating these resources and describes where you can find detailed documentation for these steps:

Analyzing Your Application

Relevant for: GUI tests only

The first step in creating a test is to analyze your application to determine your testing needs.

What development environments need to be supported by UFT?

From the perspective of UFT, your application comprises windows containing a hierarchy of objects that were created in one or more development environments. UFT provides support for these environments using add-ins.

You load UFT add-ins when UFT opens by using the Add-in Manager dialog box. You can check which add-ins are loaded by choosing **Help > About HP Unified Functional Testing**. For details, see the *HP Unified Functional Testing Add-ins Guide*.

What information does UFT need to identify objects in your application?

You need to know the URL, the executable file name and path, or other command-line information. Later, you will enter this in "Record and Run Settings Dialog Box" (described on page 865). This enables UFT to identify objects in your application and optionally open your application at the beginning of a run session. For details, see the sections describing the Record and Run options for your testing environment in the *HP Unified Functional Testing Add-ins Guide*.

What actions do you need to create?

You need to analyze the various business processes that customers perform while using your application and to create an action for each sub-process, or task, a customer might perform.

To determine which actions you need, you navigate through your application from a customer's perspective. While doing this, you perform the steps that customers might perform. Each process you perform in your application will be represented as an action in UFT. You can create your tests now, or you can wait until you are ready to add steps to your tests

As you perform a process, try to compartmentalize or "chunk" it into modular units.

Example:

An application that enables users to purchase items online might contain various business processes, including registering on the site and purchasing items. Each process may require one or more tasks—you create actions based on these tasks. For example, registering on the site may be a simple process requiring only one action, whereas purchasing items may be more complex, requiring several actions, such as a Login action, a Browse action, an AddToCart action, a PurchaseItems action, and a Logout action.

By creating separate reusable actions for each sub-process, you can include calls to the same actions from multiple tests. For example, you may want to include a Login action in many of your tests.

You can create empty actions now to set up a skeleton infrastructure for your tests, or you can create them when you are ready to add steps to your actions. For details, see "Actions in GUI Testing - Overview" on page 872.

You may also want to create a single test storing all actions relevant for an application. Then all other tests can call the actions stored in this central repository. This helps with test structure and maintenance.

Tip: As you plan your tests and actions, keep in mind that short tests and actions that check specific functions of the application or complete a transaction are better than long ones that perform several tasks, as they are easier to reuse and maintain over time.

Setting Up Object Repositories

Relevant for: GUI tests only

In this step, you build one or more object repositories and ensure that all objects have clear names that follow any predetermined naming conventions defined by your organization.

You can create object repositories using UFT functionality to recognize and learn the objects in your application, or you can manually define objects. The object repository should contain all the objects that are relevant for the tests using this infrastructure.

By creating and populating shared object repositories that can be associated with multiple actions, you can use the same object repository in multiple tests. By maintaining all objects that are relevant to an area of an application within one shared object repository, and by associating that object repository with all relevant actions, changes to the application can be reflected in the object repository without the need to update tests.

Before you create a new object repository, verify whether an object repository containing the objects you are testing already exists. If not, you can create a new object repository or add objects to an existing one.

Creating shared object repositories for the test automation infrastructure includes one or more of the following:

Changing the way that UFT identifies specific objects

This is particularly helpful when your application contains objects that change frequently or are created using dynamic content, for example, from a database. This needs to be done before you create your object repository. For details, see "Configuring Object Identification" on page 1310.

Deciding how to organize your object repositories

For individual tests, you can work with the individual action's object repositories, or you can work with a common (shared) object repository that can be used with multiple actions and multiple tests. If you are new to testing, you may want to use the default (local) object repository that is unique to each action. As you feel more comfortable with the basics of test design, you may want to take advantage of shared object repositories.

If you decide to work with shared object repositories, you need to determine how many shared object repository files are required for your application. You also need to determine which shared object repository will be used for each area of your application.

For details, see "Managing Test Objects in Object Repositories" on page 1198.

Adding (learning) objects from your application

You can instruct UFT to learn the objects in your application according to filters that you define. For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199.

Creating new objects with easily understandable names

You may create new objects to represent objects that do not yet exist in your application. You then update the properties and values of these objects as necessary after they exist in the application. For details, see "Define New Test Object Dialog Box" on page 1240.

When you create objects, make sure that they have names that are easy for application testers to recognize and that follow any established object naming guidelines. This helps make both test creation and maintenance easier over time.

Copying or moving objects from one repository to another

For details, see "Shared Object Repositories" on page 1285.

Merging objects from local repositories to shared object repositories

Application testers may create objects that are saved in an action's local object repository. You can merge these objects into a shared object repository. You can also merge two or more existing shared object repositories. For details, see "Object Repository Merge Tool" on page 1351.

Creating Function Libraries

Relevant for: GUI tests only

Creating function libraries involves developing customized functions for the application you want to test. You may want to develop functions to test special application functionality that is not already supplied by the methods in the UFT object model. This enables you to create keywords that perform operations that are not normally available for use with a particular test object class. For example, you may need to add a worksheet to an Excel file, or to generate a text file during a run session.

It may also be useful to wrap existing methods and functions together with additional programming to create application-specific functions for testing operations or sequences that are commonly performed in your application. The functions you create will be available either as extra keywords or as replacements for built-in UFT keywords during the test creation stage.

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

When creating a function library for the test automation infrastructure, you may do the following:

- **Determine whether you need to create any user-defined functions or whether you should associate any existing function libraries with your test.**

- **Determine which keywords are needed.**

- **Develop and document business-level keywords in function libraries.** For details, see "User-Defined Functions and Function Libraries" on page 1029 and "How to Create and Manage

Function Libraries" on page 1041.

- **Create the actual functions within the function libraries.** You can do this manually, or you can use the Function Definition Generator to generate function definitions and header information. For details, see "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1051.
- **Optionally define functions as new or replacement methods for test objects.** For details, see "Registered User-Defined Functions" on page 1035.
- **Debug your function libraries.** For details, see "Debug a function in a function library" on page 1042.

Configuring UFT According to Your Testing Needs

Relevant for: GUI tests only

After you set up the test automation infrastructure, you need to configure UFT to use this infrastructure. This involves one or more of the following:

Defining your global testing preferences

You need to specify configuration settings that affect how you create and run tests in general—these settings are not test-specific. For example, you can instruct UFT to record a movie of the run session under certain conditions, and to enable other HP products to run UFT tests (for example, if you want to run your tests from ALM).

You can set global testing options using the Options dialog box (**Tools > Options**) or by inserting statements in the Editor. For details, see "Global Options - Overview" on page 523.

Creating recovery scenarios

Although not directly associated with the keyword-driven methodology, the automation experts who maintain the object repositories and function libraries also often maintain a set of recovery scenarios that all application testers can associate with their tests. Recovery scenarios instruct UFT how to proceed when a step fails. For details, see "Recovery Scenarios for GUI Testing" on page 1111.

Configuring the UFT IDE to suit your testing preferences

This enables you to easily access any needed panes, such as the canvas, the Solution Explorer pane, the Toolbox pane, or the Data pane. For details, see "UFT Window Layout" on page 182.

Building Your Tests

Relevant for: GUI tests only

You can create tests that are as simple or complex as needed. In general, it is best to create tests and actions that check just one or a few simple functions or complete a transaction rather than creating long tests and actions that perform several complex tasks or that perform many tasks.

You may do the following when creating tests and test steps:

- **Create new tests, if needed.** For details, see "Create a new standalone document" on page 136 or "Create a new document within an existing solution" on page 137.
- **Create the required actions.** For details, see "Analyzing Your Application" on page 821.
- **Insert calls to the relevant actions.** For example, if the first step in a test logs in to the application, and you already created a Login action, insert a call to that action to include it in your test. For details, see "Calls to Existing Actions and Copies of Actions" on page 875.
- **Associate your object repositories with the relevant actions.** This enables you to insert steps that perform operations on those objects. For details, see "Associated Repositories Tab (Action Properties Dialog Box)" on page 906.
- **Associate your function libraries with the relevant tests.** This enables you to use your special keywords in any of the associated tests. For details, see "How to Manage Function Library Associations" on page 1045.
- **Optionally associate recovery scenarios with your test.** For details, see "How to Manage Recovery Scenario Associations" on page 1116.

Adding Steps to Your Test Actions

Relevant for: GUI tests only

When your actions are ready, you can add steps to them. This involves one or more of the following:

Adding steps by selecting the keywords (operations) that represent the application functionality you want to test

For details, see "Keyword View" on page 919.

You can insert steps in the Keyword View, the Editor, or a combination of both. You can add steps by dragging test objects from the Toolbox pane, using the **New Step** option, using the Step Generator, entering steps manually, and so on. Make sure to fill in any missing values, as needed.

For details, see "How to Add a Standard Step to Your Test or Component" on page 925, "Types of Action and Component Steps " on page 921, and "Statement Completion in the Editor" on page 306.

Enhancing your tests by inserting checkpoints and output values

- You can insert checkpoints to check for differences in the text strings, objects, and tables in your application. For details, see "Checkpoints Overview" on page 1396.
- You can insert output value steps that retrieve values in your test and store them for use as input values at a different stage in the run session. For details, see "Output Values Overview" on page 1488.

Data-driving your test

You can use the Data pane to data-drive your test using different data input during subsequent run sessions. This enables you to check how your application behaves during multiple iterations of the same action during a single run session. For details, see "Data Pane" on page 221.

Replacing fixed values with parameters

When you parameterize your test, you can check how it performs the same operations with multiple sets of data, or from data stored or generated by an external source. This enables you to increase the power and flexibility of your test. For details, see "Parameterizing Object Values" on page 1526.

Running and Troubleshooting Your Tests

Relevant for: GUI tests only

When your tests are ready, you run them, view the run results, and troubleshoot your tests, as needed.

- **Before you run a test, ensure that all of the required settings are configured as needed and that the required UFT add-ins are loaded.** Make sure that your application is open to the appropriate location for the beginning of the test, or that you instructed UFT to open it for you. Additionally, make sure that the Test Settings dialog box (**File > Settings**) and Record and Run Settings dialog box (**Record > Record And Run Settings**) are configured for your test. For details, see "Running Tests and Components" on page 634.

- **After your test runs, view the run results.** Expand the nodes in the Run Results Viewer to see where steps failed and to try to understand why. For details, see the Run Results Viewer Result Details Pane (described in the *HP Run Results Viewer User Guide*).

- **Troubleshoot your test so that it runs correctly.** For example, you may need to add or modify test steps. For details, see "Maintaining and Updating GUI Tests or Components " on page 1078.

Test a Flight Application Using the Keyword-Driven Methodology - Use-Case Scenario

Relevant for: GUI tests only

The process of creating a test is actually comprised of several steps. This section walks you through the activities you might perform for each of these steps, if you were preparing a test suite for the Mercury Tours application, including:

Mercury Tours is a Web-based demo application that simulates an online flight reservation application. You can view and experiment with this demo application at http://newtours.demoaut.com.

Define the Testing Environment for the Mercury Tours Application

Relevant for: GUI tests only

Defining the testing environment includes determining which add-ins to load and the data required to open the application.

Mercury Tours is a Web application that contains a few Java applets. Therefore, we need to ensure that the UFT Web and Java Add-ins are installed and loaded.

To open the application, we need to run a URL in a Web browser. The URL is http://newtours.demoaut.com.

Analyze the Mercury Tours Application

Relevant for: GUI tests only

When analyzing the application to determine which business processes we may want to test, we can consider both the existing business processes in the application as well as functionality that is planned for the upcoming release of the application.

The business processes that should be tested for the Mercury Tours application include:

- Registering on the site
- Reserving a flight
- Viewing the itinerary of a pending reservation
- Canceling a reservation
- Updating user profile information
- Reserving hotel rooms
- Renting a car

Although the last two items above have not yet been implemented in the application we want to test, it is important to take them into account in the planning stage.

Now that we have determined the primary business processes, we should analyze each one to determine the break-down of these business processes into their reusable building-block elements (what will later become the actions of our tests).

A logical breakdown of the above business processes could be:

- **Registering on the site**
    - Open the application
    - Go to the registration page
    - Enter the required information in the form
    - Submit the form
    - Verify that the form information is valid
        - If a mandatory field did not have a value, an error message is displayed.
        - If the password and confirm password values are not the same, an error message is displayed.
        - If the username entered in the form already exists in the database, an error message is displayed.
        - Otherwise, the successful registration page is displayed.

- **Reserving a flight**
  - Open the application
  - Sign on
  - Navigate to the Flight Finder page
  - Enter the flight details
  - Enter the service class and airline preferences
  - Click Next to navigate to the next page
  - Select the departure and return flights
  - Click Next to navigate to the next page
  - Enter the passenger details
  - Verify that the form information is valid
    - If the return date is earlier than the departure date, an error message is displayed.
    - If a mandatory field was not entered, an error message is displayed.
    - Otherwise, the flight confirmation page is displayed.
- **Viewing the itinerary of a pending reservation**
  - Open the application
  - Sign on
  - Navigate to the Itinerary page
- **Canceling a reservation**
  - Open the application
  - Sign on
  - Navigate to the Itinerary page
  - Select the reservation to cancel
  - Click the **Cancel Checked Reservations** button

    - Verify
    - Successful cancellation

- **Updating user profile information**
    - Open the application
    - Sign on
    - ...

And so on for each of the remaining processes.

Comparing the sub-items in each of the business-processes helps to identify the reusable elements of each business process.

Plan and Create the Mercury Tours Test Action Repository

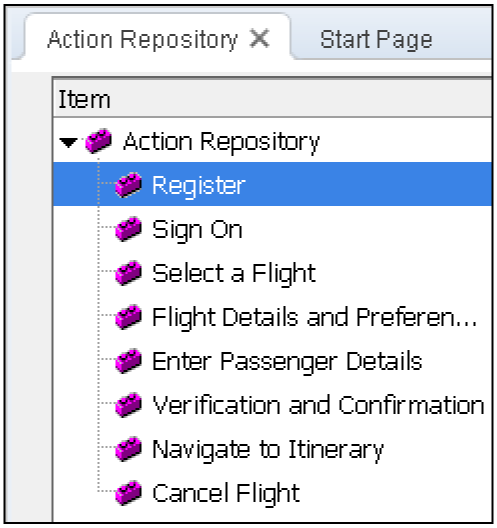
Relevant for: GUI tests only

By analyzing the breakdown performed in the previous step, we are able to identify some logical, and reusable sub-processes. Each of these is created as a reusable action.

The required actions for the set of business processes we defined could include:

- Register
- Sign On
- Flight Details and Preference
- Select a Flight
- Enter Passenger Details
- Verification and Confirmation
- Navigate to Itinerary
- Cancel Flight

Although we are not yet ready to create the actual tests or steps yet, we can create a single test. In the test, we can already define empty test actions for each of these. This test then acts as the **action repository**, and the tests that test each of our business processes all call actions from this action repository test.



Set Up the Object Repositories for the Mercury Tours Application

Relevant for: GUI tests only

Now that we know which business processes and sub-processes we want to test, we can analyze the application in detail to determine which objects are important to test and how we want to organize the objects we will learn for these tests.

We know that it is best to create manageable-sized object repositories that are organized by areas of the application.

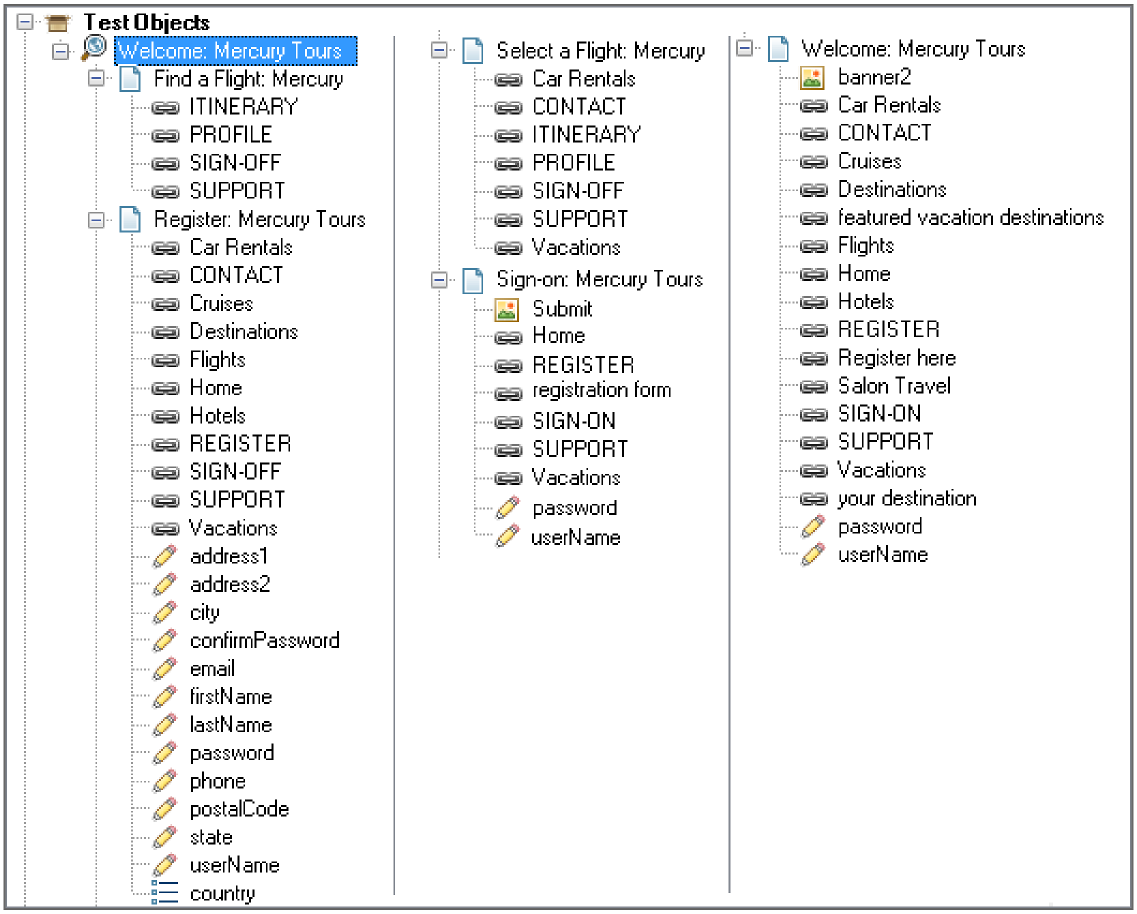
Most of the business processes we plan to test are in the central flight reservation area of the application and thus many of the same objects will be used in each of the relevant tests, but the sign on and registration processes are more standalone areas and it makes sense to store their objects separately. Thus it seems logical to create two object repository files:

- SignOn_Register
- Reservations

To create each of these repositories, we take advantage of the "Navigate and Learn Toolbar" (described on page 1302), which enables us to navigate to each page that is relevant for the object repository to automatically learn all the objects in the page. By using the filter options in the Navigate and Learn toolbar, we can ensure that we learn only the types of objects we need. For example, we can avoid learning all the non-link image objects on every page, since these objects probably do not need to be tested and would otherwise result in a larger and less manageable object repository.

Afterward, we should open the object repository for editing to delete specific objects that are not necessary and to rename objects that may otherwise be difficult to recognize when we later want to create steps with these objects.

Our **SignOn_Register** object repository may look something like this:



- Test Objects
  - Welcome: Mercury Tours
    - Find a Flight: Mercury
      - ITINERARY
      - PROFILE
      - SIGN-OFF
      - SUPPORT
    - Register: Mercury Tours
      - Car Rentals
      - CONTACT
      - Cruises
      - Destinations
      - Flights
      - Home
      - Hotels
      - REGISTER
      - SIGN-OFF
      - SUPPORT
      - Vacations
      - address1
      - address2
      - city
      - confirmPassword
      - email
      - firstName
      - lastName
      - password
      - phone
      - postalCode
      - state
      - userName
      - country
    - Select a Flight: Mercury
      - Car Rentals
      - CONTACT
      - ITINERARY
      - PROFILE
      - SIGN-OFF
      - SUPPORT
      - Vacations
    - Sign-on: Mercury Tours
      - Submit
      - Home
      - REGISTER
      - registration form
      - SIGN-ON
      - SUPPORT
      - Vacations
      - password
      - userName
    - Welcome: Mercury Tours
      - banner2
      - Car Rentals
      - CONTACT
      - Cruises
      - Destinations
      - featured vacation destinations
      - Flights
      - Home
      - Hotels
      - REGISTER
      - Register here
      - Salon Travel
      - SIGN-ON
      - SUPPORT
      - Vacations
      - your destination
      - password
      - userName

Note that each page contains only the relevant objects for the Sign on and Register business processes.

Create the Function Libraries and Functions Required for Testing the Mercury Tours Application

Relevant for: GUI tests only

In some of our business processes, we want to test not only that the business processes can be performed to completion, but that certain features in the application behave as expected.

Because testing such functionality requires complex programming, and because we want to test the functionality in several different sub-processes, it makes sense to create these functionality checks in the form of functions, and to store them in function libraries, so that we can call the functions from more than one test action.

For example, we want to verify that the Mercury Tours application properly handles various invalid data in forms and we want to verify that the application properly calculates ticket prices for various types of itineraries.

We also want to make sure that we have ways to recover from certain application problems so that if such a problem occurs while a step is running, it does not prevent the action or test from completing its run or prevent other tests from running afterward. This recovery function can be used by recovery scenarios that we will associate with our tests at later stages.

At this stage, we can create a function library containing functions such as:

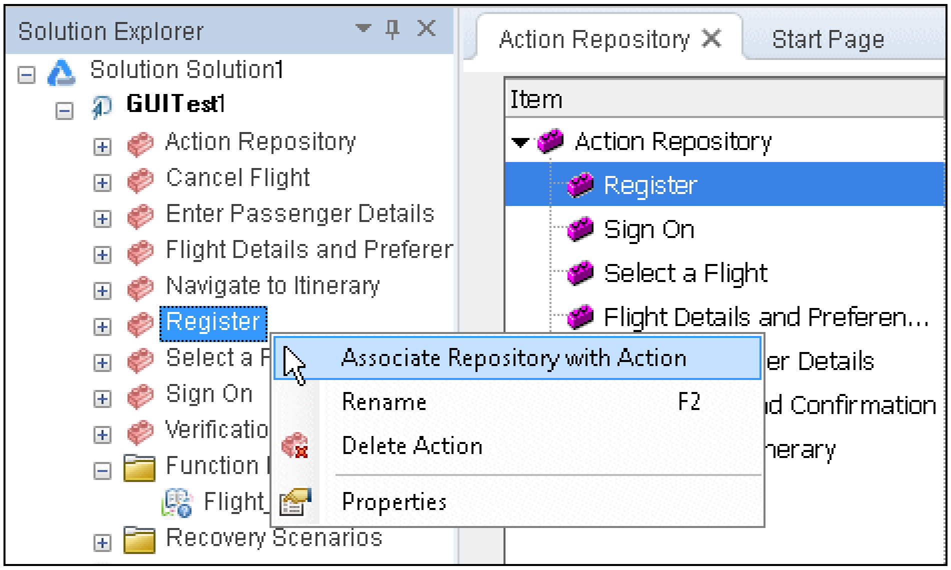
- VerifyForm
- VerifyTicketPrice
- DataBaseFailureRecoveryFunction

Create Tests and Test Steps for the Mercury Tours Business Processes

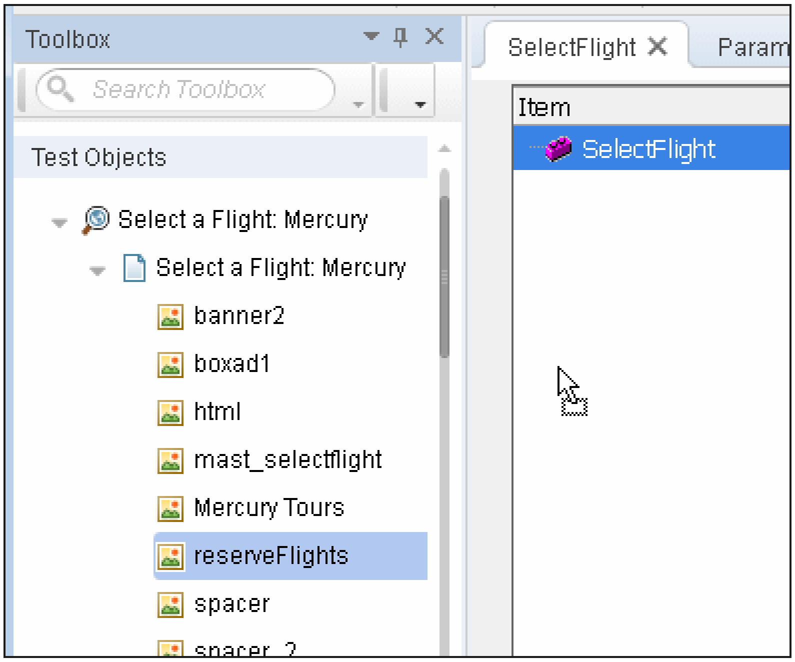
Relevant for: GUI tests only

Now that we have planned and prepared all of the required resources for our tests, we are ready to use them to create tests and test steps that represent the steps a real user would perform on the Mercury Tours application as well as inserting functions that verify the expected functionality of various features.

We start by using the solution explorer to associate the relevant object repository with each action in the action repository test and to associate our function library with the test as well.



Then we use the Toolbox pane to drag objects and functions into our actions to create the individual steps of each action.



As we design our steps, we make sure to parameterize method arguments as necessary to maximize reusability of the actions in different business processes (tests).

Finally, we create new tests for each of the processes we defined in the "Analyze the Mercury Tours Application" step (see page 828). We use the solution explorer to associate our function library with each test and then we insert calls to the relevant actions.

Tasks

How to Create a GUI Test Using the Keyword-Driven Methodology

Relevant for: GUI tests only

This task describes how to create a test using the keyword-driven methodology. For details on a specific step, see the parallel section in "Keyword-Driven Methodology - Overview" on page 821.

This task includes the following steps:

- "Analyze your application" below
- "Prepare the testing infrastructure " below
- "Add steps to the actions in your test action repository" on page 838
- "Enhance your test " on page 838
- "Run your test " on page 839
- "Analyze the run results and report any defects" on page 840

1. Analyze your application

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

- **Determine the development environments in which your application controls were developed**, such as Web, Java, or .NET, so that you can load the required UFT add-ins.
- **Determine the functionality that you want to test.** To do this, you consider the various activities that customers perform in your application to accomplish specific tasks. Which objects and operations are relevant for the set of business processes that need to be tested? Which operations require customized keywords to provide additional functionality?
- **Decide how to divide these processes into smaller units that will be represented by your test's actions.** Each action should emulate an activity that a customer might perform when using your application.

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

For an overview of this step, see "Analyzing Your Application" on page 821.

2. Prepare the testing infrastructure

To complete the infrastructure that is part of the planning process, you need to:

- **Build the set of resources** to be used by your tests, including:
  - **shared object repositories** containing test objects (which are representations of the objects in your application).

    For details, see "Managing Test Objects in Object Repositories" on page 1198.
  - **function libraries** containing functions that enhance UFT functionality.

    For details, see "User-Defined Functions and Function Libraries" on page 1029
  - **recovery scenarios** that instruct UFT to recover from unexpected events and errors that occur in your testing environment during a run session. (Optional)

    For details, see "Recovery Scenarios for GUI Testing" on page 1111.
  - **additional optional files**, such as data table files and environment variable files.
- **Configure UFT according to your testing needs.** This can include:
  - setting up your **global testing preferences**.

    For details, see "Global Options - Overview" on page 523.
  - setting up any **test-specific preferences**.

    For details, see "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.
  - configuring your **run session preferences**.

    For details, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 526, "Test Runs Pane (Options Dialog Box > GUI Testing Tab)" on page 539, and "Run Pane (Test Settings Dialog Box)" on page 601.
  - defining and associating **recovery scenarios**.

    For details, see "Recovery Scenarios for GUI Testing" on page 1111.
  - creating **automation scripts** that automatically set the required configurations (such as the add-ins to load) on the UFT client at the beginning of a run session. For details, see "UFT Automation Scripts" on page 1155.
- **Create one or more tests that serve as action repositories** in which you can store the actions to be used in your tests, as this enables you to maintain your actions in one central location.

- **Associate your shared object repositories with the relevant actions** (in these action repositories). This enables you to later insert steps using the objects stored in the object repositories. When you create your tests, you insert calls to one or more of the actions stored in this repository.

  For an overview of this step, see "Setting Up Object Repositories" on page 823.

  For details, see "Associate Repositories Dialog Box" on page 1266.

3. **Add steps to the actions in your test action repository**

- **(Prerequisite) Associate your function libraries and recovery scenarios with the relevant tests, so that you can insert steps using keywords.**

  For details, see "Solution Explorer Pane User Interface" on page 478.

- **Create steps using keyword-driven functionality**. You can use the table-like, graphical Keyword View—or you can use the Editor if you prefer to program steps directly in VBScript.

  You can add steps to your test in one or both of the following ways:

  - **Drag objects from your object repository or from the Toolbox pane** to add keyword-driven steps in the Keyword View or the Editor.

    The object repository and Toolbox pane contain all of the objects that you want to test in your application. You created one or more object repositories when you prepared the testing infrastructure, as described in "Prepare the testing infrastructure " on page 836.

  - **Record on your application.** As you navigate through your application during a recording session, each step you perform is graphically displayed as a row in the Keyword View. For details, see "Keyword View" on page 919.

  For an overview of this step, see "Creating Function Libraries" on page 824.

4. **Enhance your test**

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

- **Insert checkpoints and output values** into your test.

  For details, see "Checkpoints Overview" on page 1396 and "Output Values Overview" on page 1488.

- **Replace fixed values with parameters** to broaden the scope of your test. For details, see "Parameterizing Object Values" on page 1526.

- **Add user-defined functions** by creating function libraries and calling their functions from your test.

  For details, see "User-Defined Functions and Function Libraries" on page 1029.

- **Use the many functional testing features** included in UFT to enhance your test and/or add programming statements to achieve more complex testing goals.

  For details, see "Generated Programming Operations" on page 968.

For an overview of this step, see:

- "Creating Function Libraries" on page 824
- "Configuring UFT According to Your Testing Needs " on page 825
- "Building Your Tests" on page 825
- "Adding Steps to Your Test Actions" on page 826

5. ## Run your test

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

- **Run your test to check your application.** The test starts running from the first line in your test and stops at the end of the test. While running, UFT connects to your application and performs each operation in your test, including any checkpoints, such as checking any text strings, objects, tables, and so on. If you parameterized your test with data table parameters, UFT repeats the test (or specific actions in your test) for each set of data values in the Data pane.
  For details, see "Running Tests and Components" on page 634.

- **Run your test to debug it. (Optional)**

  You can:

  - Control your run session to help you identify and eliminate defects in your test.
  - Use the **Step Into**, **Step Over**, and **Step Out** commands to run your test step by step.
  - Begin your run session from a specific step in your test, or run the test until a specific step is reached.
  - Set breakpoints to pause your test at predetermined points.
  - View or change the value of variables in your test each time it stops at a breakpoint in the Debug Viewer.

- Manually run VBScript commands in the Debug Viewer.

  For details, see "Debugging Tests and Components" on page 674.

- **Run your test using Maintenance Run Mode after your application changes. (Optional)**

  Use **Maintenance Run Mode** to update your test when you know that your application has changed, and you therefore expect that UFT will not be able to identify the objects in your test. For details, see "Maintaining and Updating GUI Tests or Components " on page 1078.

- **Run your test using Update Run Mode to update one or more of the following: (Optional)**

  - property sets used for test object descriptions
  - expected checkpoint values
  - data available to retrieve in output values
  - Active Screen images and values

  For an overview of this step, see "Running and Troubleshooting Your Tests" on page 827.

6. Analyze the run results and report any defects

After you run your test, you can:

- **View the results of the run in the Run Results Viewer.** You can view a summary of your results as well as a detailed report.

  - If you captured still images or movies of your application during the run, you can view these from the Screen Recorder tab of the Run Results Viewer.

    For details, see the Screen Recorder Pane (described in the *HP Run Results Viewer User Guide*).

  - If you enabled local system monitoring for your test, you can view the results in the System Monitor tab of the Run Results Viewer.

    For details, see the section on the System Monitor Pane (described in the *HP Run Results Viewer User Guide*).

- **Report defects detected during a run session. (Optional)**

  If you have access to ALM, you can report the defects you discover to the project database in either of the following ways:

  - Instruct UFT to automatically report each failed step in your test.

- Report failed steps manually from the Run Results Viewer.

For details, see "ALM Integration" on page 707.

Chapter 32: Recording GUI Tests and Components

Relevant for: GUI tests and components

This chapter includes:

Concepts

Recording GUI Tests and Components - Overview

Relevant for: GUI tests and components

You can create the main body of a test or component by recording the typical processes that users perform on your application. UFT records the operations you perform, adding them as steps to the selected test action or component.

A step is anything you do that changes the content of a page or object in the application, for example, clicking a link or typing data in an edit box. The steps in your action or component represent the operations you perform on your application. As you record the action or component, if it is visible in the Keyword View or the Editor, you can see the steps that are added based on the operations you perform.

During a run session, UFT uses the recorded steps to replicate the operations you performed while recording.

While you record the steps, UFT creates test objects representing the objects in your application on which you perform operations, and stores them in an object repository. This enables UFT to identify the objects in your application both while creating the test or component and during a run session.

Recording can be useful in the following circumstances:

- You are new to UFT and want to learn how UFT interprets the operations you perform on your application and how it converts them to UFT objects and built-in operations.
- You need to quickly create a test or component that tests the basic functionality of an application or feature, and does not require long-term maintenance.
- You are working with a new application or with major new features of an existing application, and you want to learn how UFT interacts with the application.
- You are developing functions that incorporate built-in UFT keywords.

When you record a test or component, UFT enters the correct objects, methods, and argument values for you. Therefore, it is possible to create a basic test or component with little preparation or planning. During a recording session, you can also add checkpoint and output value steps, to check or retrieve values from your application.

When recording a test, you can use the standalone Record toolbar, to divide the steps recorded into multiple actions. For details on the Record toolbar, see "Record Toolbar" on page 866. For details on why and how to work with multiple actions, see "Actions in GUI Testing" on page 871.

By default, UFT records in the normal recording mode. For tests and scripted components, UFT supports additional recording modes that you can use. For example, these recording modes can be useful in the following situations:

- If you are unable to record on an object in a given environment in the standard recording mode.
- If you want to record mouse clicks and keyboard input with the exact x- and y-coordinates.
- If you want UFT to recognize objects in your application based on what they look like, instead of using properties that are part of their design.

For details, see "Recording Modes" on the next page.

You can also create a test or component by using the keyword-driven methodology, which enables you to select keywords to indicate the operations you want to perform on your application, as described in "Test Creation - Keyword-Driven Methodology" on page 820.

Guidelines for Recording Tests and Components

Relevant for: GUI tests and components

- Before you begin recording, you need to ensure that your tests and components cover your testing requirements. For details on planning your tests and components, see "GUI Test Creation Overview" on page 812.
- Consider increasing the power and flexibility of your test or component by replacing fixed values with parameters. When you parameterize your test or component, you can check how it performs the same operations with multiple sets of data, or from data stored or generated by an external source. For details, see "Parameterizing Object Values" on page 1526.
- When you record tests or components, you may not notice that new objects are being added to the local object repository. This may result in many testers maintaining local object repositories with copies of the same objects. When using a keyword-driven methodology, you select the objects for your steps from the existing object repository. When you need a new object, you can add it to your local object repository temporarily, but you are also aware that you need to add it to the shared object repository for future use.
- If you are recording steps on a Web-based application, evaluate the types of events you need to record. If you need to record more or fewer events than UFT generally records by default, you can configure the events you want to record. For details, see the section on configuring Web event recording in the *HP Unified Functional Testing Add-ins Guide*.
- If you are creating a test or component on Web objects, you can record your test or component on Microsoft Internet Explorer or Mozilla Firefox and run it on another supported browser (according to the guidelines specified in the *HP Unified Functional Testing Product Availability Matrix*, available from the UFT Help or the root folder of the Unified Functional Testing DVD).

  UFT supports running tests and components on the following browsers—Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, and applications with embedded Web browser controls. UFT can also connect to a remote Mac computer (with a UFT Connection Agent installed) and run tests and components on Safari. For details, see the *HP Unified Functional Testing Add-ins Guide*.

- If you have objects that behave like standard objects, but are not recognized by UFT, you can define your objects as virtual objects. For details, see "How to Define Virtual Objects for Unsupported Objects in Your Test or Scripted Component" on page 1387.
- After you create your test or component, you can enhance it using checkpoints and other special testing options. After creating your initial test, you can further enhance it by adding and modifying steps in the Keyword View or Editor.

Recording Modes

Relevant for: GUI tests and scripted GUI components

UFT provides the following recording modes:

- **Normal Recording.** Records the objects in your application and the operations performed on them. This mode is the default and takes full advantage of the UFT test object model, recognizing the objects in your application regardless of their location on the screen. For details, see "How to Record a GUI Test" on page 853.

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

- **"Analog Recording".** Enables you to record the exact mouse and keyboard operations relative to the screen or the application window. This method records mouse movements. For details, see "Analog Recording" on the next page.
- **"Low-Level Recording".** Enables you to record operations on the exact coordinates of any object, whether or not UFT recognizes the specific object or the specific operation. This method does not record mouse movements. For details, see page "Low-Level Recording" on page 847.
- **"Insight Recording" (tests and scripted components only).** Enables you to record operations on objects that UFT recognizes based on their appearance, instead of their native properties. This can be useful if you are recording on an application whose technology is not supported by UFT, or on an application running on an emulator or a remote computer. For details, see page "Insight Recording" on page 849.

Use analog, low-level, or Insight recording only when normal recording mode does not accurately record your operation. Analog, low-level, and Insight recording require more disk space than normal recording mode.

> **Note:** UFT supports an additional recording mode, Standard Windows recording, which is relevant when recording tests or components on SAP GUI for Windows applications. For details, see the SAP GUI for Windows section of the *HP Unified Functional Testing Add-ins Guide*.

Analog Recording

Relevant for: GUI tests and scripted GUI components

This method enables you to record the exact mouse and keyboard operations that you perform, in relation to either the screen or the application window. In this recording mode, UFT records and tracks every movement of the mouse as you drag the mouse around a screen or window.

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

Use analog recording for applications in which the actual movement of the mouse is what you want to record. These can include drawing a mouse signature or working with drawing applications that create images by dragging the mouse.

You can switch to analog recording in the middle of a recording session for specific steps. After you record the necessary steps using analog recording, you can return to normal recording mode for the remainder of your recording session.

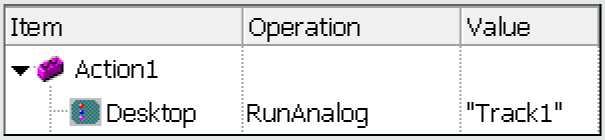
For task details, see "How to Record Using Analog Recording" on page 858.

Considerations for Analog Recording

- You can record in analog recording mode relative to the screen or relative to a specific window. For details, see "Analog Recording Settings Dialog Box " on page 863.
- The steps recorded using analog recording are saved in a separate data file. This file is stored with the action in which the analog steps are recorded.
- When you record in analog recording mode, UFT adds to your test action or scripted component a single **RunAnalog** statement that calls the recorded analog file. The corresponding Active Screen displays the results of the last analog step that was performed during the analog recording session.
- You cannot edit analog recording steps from within UFT.
- Analog recording requires more disk space than normal recording mode.

Example

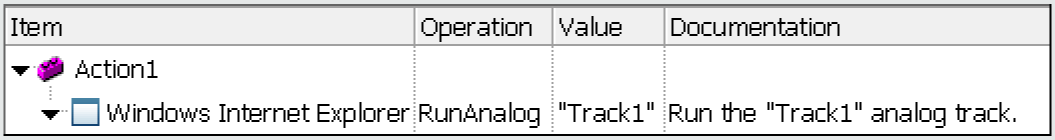
If you chose to **Record relative to the screen**, UFT inserts the **RunAnalog** step for a Desktop item. For example:



Item	Operation	Value
Action1		
Desktop	RunAnalog	"Track1"

```
Desktop.RunAnalog "Track1"
```

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



Item	Operation	Value	Documentation
Action1			
Windows Internet Explorer	RunAnalog	"Track1"	Run the "Track1" analog track.

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

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

To use this track file in other actions or tests, you must call the action or scripted component that contains the **RunAnalog** step.

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

To stop an analog step in the middle of a run session, press Ctrl + Esc, then click **Stop** in the Testing toolbar.

Low-Level Recording

Relevant for: GUI tests and scripted GUI components

This method enables you to record on any object in your application, whether or not UFT recognizes the specific object or the specific operation. This mode records at the object level and records all run-time objects as Window or WinObject test objects.

You can switch to low-level recording in the middle of a recording session for specific steps. After you record the necessary steps using low-level recording, you can return to normal recording mode for the remainder of your recording session.

For task details, see "How to Record Using Low-Level Recording" on page 859.

Considerations for Low-Level Recording

- Use low-level recording for recording on environments or objects not supported by UFT, if the appearance of the objects might change, but their location will not. If the object's appearance will not change, you can use Insight recording for unsupported environments or objects.

- Use low-level recording for when you need to record the exact location of the operation on your application screen. While recording in normal mode, UFT performs the step on an object even if it has moved to a new location on the screen. If the location of the object is important to your test or scripted component, switch to low-level recording to enable UFT to record the object in terms of its x- and y- coordinates on the screen. This way, the step will pass only if the object is in the correct position.

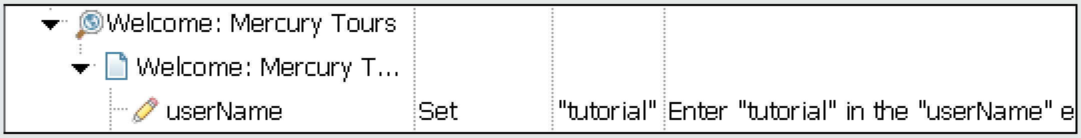
- While low-level recording, UFT records all parent level objects as Window test objects and all other objects as WinObject test objects. They are displayed in the Active Screen as standard Windows objects.

- Low-level recording supports the following methods for each test object:
  - WinObject test objects: **Click**, **DblClick**, **Drag**, **Drop**, **Type**
  - Window test objects: **Click**, **DblClick**, **Drag**, **Drop**, **Type**, **Activate**, **Minimize**, **Restore**, **Maximize**
- Each step recorded in low-level recording mode is added to the test and can be viewed in the Keyword View and Editor. (Analog recording records only the one step that calls the external analog data file.)
- Low-level recording mode is not fully supported for multibyte character input.
- Steps recorded using low-level recording mode may not run correctly on all objects.
- Low-level recording requires more disk space than normal recording mode.

Example

The following examples illustrate the difference between the same operations recorded using normal mode and low-level recording mode.

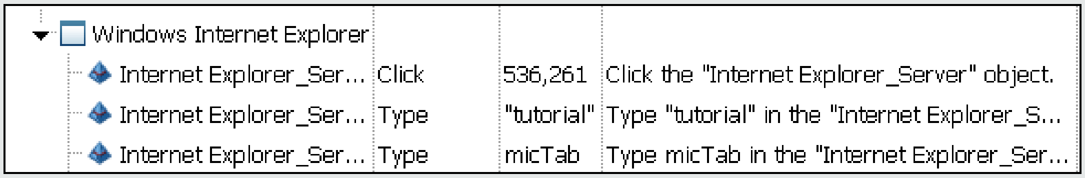
Suppose you type the word tutorial into a user name edit box and then press the TAB key while in normal recording mode. Your action or scripted component is displayed as follows in the Keyword View and Editor:



Welcome: Mercury Tours			
Welcome: Mercury T...			
userName	Set	"tutorial"	Enter "tutorial" in the "userName" e

Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "tutorial"

If you perform the same action while in low-level recording mode, UFT records the click in the user name box, followed by the keyboard input, including the TAB key. Your action or scripted component is displayed as follows in the Keyword View and Editor:



Windows Internet Explorer			
Internet Explorer_Ser...	Click	536,261	Click the "Internet Explorer_Server" object.
Internet Explorer_Ser...	Type	"tutorial"	Type "tutorial" in the "Internet Explorer_S...
Internet Explorer_Ser...	Type	micTab	Type micTab in the "Internet Explorer_Ser...

Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Click 536,261
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Type "tutorial"
Window("Windows Internet Explorer").WinObject("Internet Explorer_Server").Type micTab

Insight Recording

Relevant for: GUI tests and scripted GUI components

This mode enables you to record on any control (object) displayed on your screen, whether or not UFT recognizes the object's technology and is able to retrieve its properties or activate its methods.

In this mode, UFT recognizes controls based on their appearance, and not their native properties. This can be useful to test controls from an environment that UFT does not support or even from a remote computer running a non-Windows operating system.

You can switch to Insight recording in the middle of a recording session for specific steps. After you record the necessary steps using Insight recording, you can return to normal recording mode for the remainder of your recording session.

For task details, see "How to Record a Test or Component Using Insight Recording" on page 860.

Considerations for Insight Recording

- Use Insight recording for recording on applications, environments, or controls that UFT does not recognize when using the normal recording mode.

- While recording in the Insight recording mode, UFT creates an InsightObject test object for every control on which you perform an operation. For details, see "Identifying Objects Using Insight" on page 1176.

  With every InsightObject, UFT stores an image of the control that the test object represents, and, optionally, a number of snapshots of the application containing the control.

  After you record the objects, you can use these snapshots in the Change Test Object Image dialog box (described on page 1233) to change the image stored with the test object in the object repository, if necessary.

- If the **Save the clicked coordinates as the test object's ClickPoint** option is selected in the Insight pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Insight** node), the test object's ClickPoint is set to the first location that you clicked on the control. You can modify the ClickPoints manually in the "Change Test Object Image / Add Insight Test Object Dialog Box" (described on page 1233) dialog box when you finish recording.

  If you want the ClickPoint for Insight objects set to **Center** when they are added during a recording session, clear the **Save the clicked coordinates as the test object's ClickPoint** option.

- When created, all Insight test objects are named **InsightObject**, with an incremental suffix added if necessary, to avoid duplicate test object names within a single parent test object.

  After recording, you can modify the recorded test objects and steps to improve the readability and efficiency of your test or component. For details, see the step about fine-tuning recorded objects in "How to Record a Test or Component Using Insight Recording" on page 860.

- Each step recorded in Insight recording mode is added to the test and can be viewed in the Keyword View and Editor. In the Editor, the test object image is displayed in the step instead of the test object name. (To show test object names instead, clear the **Show test object image in steps** in the Insight pane (**Tools > Options > GUI testing** tab **> Insight** node.)

- Insight recording requires more disk space than normal recording mode, because of the test object image and the snapshots stored with the test object.

  To control the amount of space used, you can adjust the number of snapshots saved and their size in the "Insight Pane (Options Dialog Box > GUI Testing Tab)" described on page 561. These settings can also affect UFT performance when recording and running Insight steps.

  In addition, after you finish recording all of the relevant test objects, and modifying them to your satisfaction in the object repository, you can delete all of the snapshots to reduce the amount of disk space used. This does not delete the test object's image, which is used to recognize the object in the application.

Troubleshooting for Insight Recording

- When using Insight recording to record operations on a remote computer accessed by a program other than Remote Desktop Connection, the boundaries of the control being learned might not be recognized correctly. This happens because unlike Remote Desktop Connection, other programs do not suppress the cursor during screen capture. The cursor's presence in the capture makes the boundaries of the control more difficult to recognize.

  Workaround: Configure the remote access program to not display the remote cursor. Alternatively, you can modify the test object image saved with the learned test object after the recording session. For details, see "Change Test Object Image / Add Insight Test Object Dialog Box" on page 1233.

- The Insight recording technology is based on image processing. During a recording session, clicking on control images that contain straight lines may result in the object being recorded incorrectly. Make sure that you click in the center of such objects.

- During a recording session, when recording a mix of keyboard and mouse events (typing and clicking) using Insight recording, the order of the recorded steps might not reflect the actual order of events.

  Only use Insight recording for mouse events. When possible, keyboard events should be recorded using standard or low-level recording.

  If you have to record keyboard events in Insight, wait two seconds before and after typing to ensure that the events are recorded in the correct order.

  If necessary, you can change the order of the steps manually after recording.

- When recording in Insight mode, window operations such as Close, Move, Minimize, and Maximize may be recorded on the wrong parent object.

  This is because in Insight mode, UFT processes the user's operations without halting the application's responses. The image-processing algorithms used to identify controls are time consuming, and in some cases, the operation is processed and recorded after the application carried it out. If the window already closed or moved, the operation is recorded on the object that is currently present in the clicked location.

  Workaround: Do one of the following:

  - Use standard recording to record these operations.
  - Instead of recording the operation, learn the control (in the Object Repository window, use **Add Insight Object to Local**) and compose a step that clicks on the object.

- When recording in Insight mode, drag and drop operations of top-level windows may not be recorded.

  Workaround: Do one of the following:

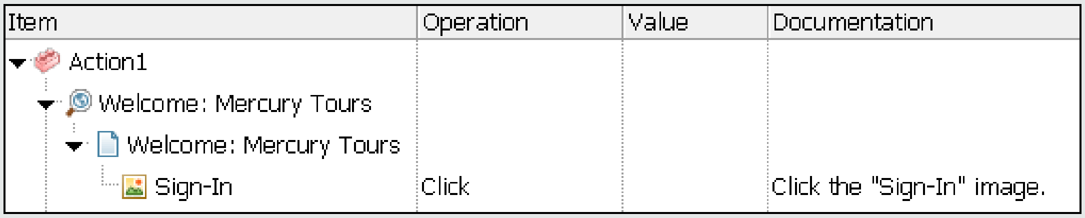
  - Record the drag and drop operations using normal recording. UFT will record these steps on regular, non-Insight, UFT test objects.
  - Learn the top-level window as a regular UFT test object and then compose the Drag and Drop steps in the editor.

- **Dual monitor support.** When recording in Insight mode, UFT records only on the primary monitor. Therefore, if you are working with dual monitors, make sure that your application is visible on the primary monitor during a recording session.

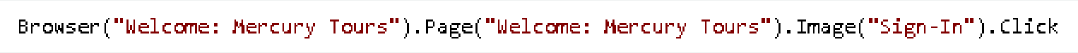
Insight Recording Example

The following examples illustrate the difference between the same operation recorded using normal mode and Insight recording mode.

Suppose you click the **Sign-in** button on the Mercury Tours Web site while in normal recording mode. UFT recognized the Web control, and the recorded step is displayed as follows in the Keyword View and Editor:

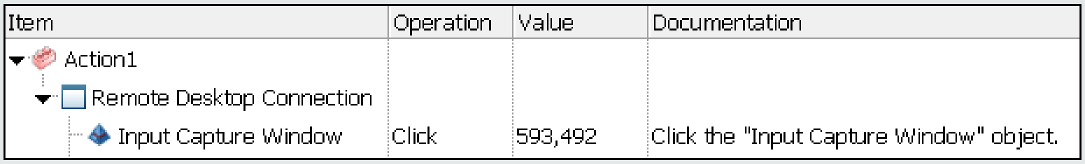


Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
Sign-In	Click		Click the "Sign-In" image.

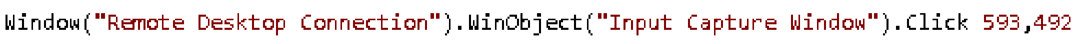


```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click
```

However, if you perform the same action on a browser running on a remote computer, UFT does not recognize the button as a Web control. Instead, UFT recognizes only the Remote Desktop Connection window as a control, and your click is recorded at the relevant coordinates within it. The recorded step is displayed as follows in the Keyword View and Editor:



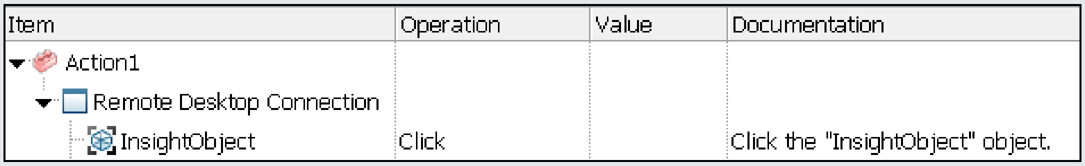
Item	Operation	Value	Documentation
Action1			
Remote Desktop Connection			
Input Capture Window	Click	593,492	Click the "Input Capture Window" object.



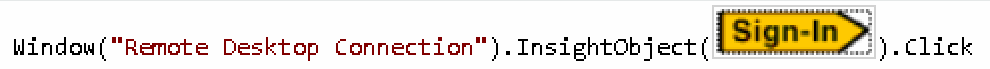
```
Window("Remote Desktop Connection").WinObject("Input Capture Window").Click 593,492
```

For this step to succeed during a run session, the browser window must be in the exact same location within the Remote Desktop Connection window.

You can solve this problem by recording the step using Insight recording. The recorded step is displayed as follows in the Keyword View and Editor:



Item	Operation	Value	Documentation
Action1			
Remote Desktop Connection			
InsightObject	Click		Click the "InsightObject" object.



```
Window("Remote Desktop Connection").InsightObject(Sign-In).Click
```

This step is easier to read, and will succeed even if the browser window is located in a different location within the remote desktop window. This is because UFT looks for a shape that resembles the **Sign-in** button anywhere inside the Remote Desktop Connection control.

Tasks

How to Record a GUI Test

Relevant for: GUI tests only

This task describes how to create a test in UFT by recording the steps that you perform on your application.

This task includes the following steps:

- "General Prerequisites" below
- "Web-based Applications Prerequisites" below
- "Start a recording session and set the Record and Run settings if necessary" on the next page
- "Record steps into the test" on the next page
- "Use the Record toolbar to manage your recording session" on the next page
- "Switch to other recording modes - optional" on page 855
- "Stop recording and save the test" on page 855

1. General Prerequisites

- **Close unnecessary applications**. Before you start to record, close all applications not required for the recording session.

- **Determine application access**. Decide how you want to open the application when you record and run your test. You can choose to have UFT open one or more specified applications, or record and run on any application that is already open. The "Record and Run Settings Dialog Box" (described on page 865) contains tabbed pages corresponding to the add-ins loaded. For details, see the section on setting Record and Run options in the *HP Unified Functional Testing Add-ins Guide*.

- **Set global record and run options**. Choose how you want UFT to record and run your test by setting global testing options in the Options dialog box and settings specific to your test in the Test Settings dialog box. For details, see "UFT Global Options" on page 522 and "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.

2. Web-based Applications Prerequisites

Determine Browser activation timing. UFT can record only on Web browsers that were opened after UFT. If you plan to use the **Record and run tests on any open browser** option in the "Record and Run Settings Dialog Box" (described on page 865), make sure that the browser you want to use for recording was opened after you opened UFT.

Set security zone. If you are recording on a Web site, determine the security zone of the site. When you record on a Web browser, the browser may prompt you with security alert dialog boxes. You may choose to disable/enable these dialog boxes.

Modify object values. If you are recording on a Web object, you must modify the object's value to enable UFT to record the step. For example, to record a selection in a WebList object, you must click on the list, scroll to an entry that was not originally showing, and select it. If you want to select the item in the list that is already displayed, you must first select another item in the list (click it), then return to the originally displayed item and select it (click it).

Modify the Web event configuration. If you are recording on a Web object, you may want to select a predefined configuration level in the Web Event Recording Configuration dialog box (described in the *HP Unified Functional Testing Add-ins Guide*). By default, UFT uses the **Basic** recording level. If UFT does not record all the events you need, you may require a higher recording level.

3. Start a recording session and set the Record and Run settings if necessary

a. Select the action that you want to record. To do this, do one of the following:

   - In the canvas of an open test, click the action.

   - Bring an open action document into focus.

b. Click the **Record** button or select **Record > Record**.

c. If the "Record and Run Settings Dialog Box" (described on page 865) opens, set the required settings. The settings in this dialog box can determine whether to open an application for you at the beginning of record and run settings. It can also control on which applications UFT records. For example, for performance reasons, the default setting in the Windows Applications tab is to record and run only on the applications you specify (and not on any open application). If you do not specify an application or change this option, UFT will not record or run on any Windows-based application.

   If the dialog box does not open automatically (for example, you are recording additional steps in an existing test or you manually opened and set options in this dialog box before beginning the record session), but you want to view or modify the settings, you can open it using the **Record > Record And Run Settings** dialog box menu command.

   UFT is minimized, and a standalone "Record Toolbar" (described on page 866) is displayed.

4. Record steps into the test

   Navigate through your application. UFT records each step you perform and adds it to your test.

5. Use the Record toolbar to manage your recording session

- **To add your steps to existing actions:** From the **<Action Name>** drop-down list, select the action name in which to include the performed steps.

- **To insert a call to a new action or call to an existing action:** In the Record toolbar, click the **Insert Call to New Action** button and choose the type of action call you want. In the subsequent dialog box, name the action and define its properties.

- **To insert a checkpoint/output value:** Click the **Insert Checkpoint and Output Value** button and choose the type of checkpoint or output value to insert. In the subsequent dialog box, name the checkpoint and define its properties.

6. Switch to other recording modes - optional

For details, see:

- Analog recording: "How to Record Using Analog Recording" on page 858

- Low-level recording: "How to Record Using Low-Level Recording" on page 859

- Insight recording: "How to Record a Test or Component Using Insight Recording" on page 860

- Standard Windows recording (relevant when recording on SAP GUI for Windows applications): The SAP GUI for Windows section of the *HP Unified Functional Testing Add-ins Guide*

7. Stop recording and save the test

a. When you complete your recording session, click the **Stop** button on the Record toolbar or select **Record > Stop**.

b. To save your test, click the **Save** button or select **File > Save**.

How to Record a GUI Component

Relevant for: GUI components only

This task describes how to create a component in UFT by recording the steps that you perform on your application.

This task includes the following steps:

- "General Prerequisites" on the next page
- "Web-based Applications Prerequisites" on the next page
- "Start a recording session and set the Application settings if necessary" on page 857

- "Record steps into the component " on the next page
- "Use the Record toolbar to manage your recording session" on the next page
- "Switch to other recording modes - optional (scripted components only)" on the next page
- "Stop recording and save the component" on page 858

1. **General Prerequisites**

   - **Close unnecessary applications.** Before you start to record, close all applications not required for the recording session.

   - **Set global record and run options.** Choose how you want UFT to record and run your component by setting global options in the Options dialog box and settings specific to your component in the Business Component Settings dialog box. For details, see "UFT Global Options" on page 522 and "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.

     > **Note:** The settings that determine the windows-based applications on which the component can record and run are defined in the application area (click **Additional Settings** in the application area's sidebar and select **Applications**). You can view these settings, in read-only mode, in the Business Components Settings dialog box (**File > Settings > Applications pane**) or in the Applications dialog box that opens when you begin recording. For details, see:
     >
     > - "Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 597.
     > - "Associate a different application area with your component" on page 2090.

2. **Web-based Applications Prerequisites**

   Determine Browser activation timing. UFT can record only on Web browsers that were opened after UFT. If you plan to record a web event, make sure that the browser you want to use for recording was opened after you opened UFT.

   Set security zone. If you are recording on a Web site, determine the security zone of the site. When you record on a Web browser, the browser may prompt you with security alert dialog boxes. You may choose to disable/enable these dialog boxes.

   Modify object values. If you are recording on a Web object, you must modify the object's value to enable UFT to record the step. For example, to record a selection in a WebList object, you must click on the list, scroll to an entry that was not originally showing, and select it. If you want to select the item in the list that is already displayed, you must first select another item in the list (click it), then return to the originally displayed item and select it (click it).

Modify the Web event configuration. If you are recording on a Web object, you may want to select a predefined configuration level in the Web Event Recording Configuration dialog box (described in the *HP Unified Functional Testing Add-ins Guide*). By default, UFT uses the **Basic** recording level. If UFT does not record all the events you need, you may require a higher recording level.

3. Start a recording session and set the Application settings if necessary

a. Click the **Record** button or select **Record > Record**.

b. If this is the first time that you are recording in the current UFT session, the Applications dialog box opens.

Verify that the application area is defined with the settings you need for your recording session. If the settings do not meet your requirements, you may need to associate a different application area with your component. For details, see "Associate a different application area with your component" on page 2090.

> **Note:** If the Applications dialog box does not open, you can also verify the settings in the Applications pane of the Settings dialog box (**File > Settings**), which has the same user interface elements. For details, see "Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 597.

UFT is minimized, and a standalone Record toolbar is displayed. For user interface details on the Record toolbar, see "Record Toolbar" on page 866.

4. Record steps into the component

Navigate through your application. UFT records each step you perform and adds it to your component.

5. Use the Record toolbar to manage your recording session

To insert a checkpoint/output value: Click the **Insert Checkpoint and Output Value** button and choose the type of checkpoint or output value to insert. In the subsequent dialog box, name the checkpoint and define its properties.

6. Switch to other recording modes - optional (scripted components only)

For details, see:

- Analog recording: "How to Record Using Analog Recording" on the next page
- Low-level recording: "How to Record Using Low-Level Recording" on page 859
- Insight recording: "How to Record a Test or Component Using Insight Recording" on page 860

- Standard Windows recording (relevant when recording on SAP GUI for Windows applications): The SAP GUI for Windows section of the *HP Unified Functional Testing Add-ins Guide*

7. **Stop recording and save the component**

   a. When you complete your recording session, click the **Stop** button on the Record toolbar or select **Record > Stop**.

   b. Save your component by clicking the **Save** button or selecting **File > Save**.

How to Record Using Analog Recording

Relevant for: GUI tests and scripted GUI components

This task describes how to record a test or scripted component using analog recording, in which your keyboard input, mouse movements, and clicks are recorded and saved in an external data file.

When UFT runs the test or scripted component, this external data file is called, and every movement, mouse click, and operation are replicated exactly as you recorded them.

> **Note:** This task is part of a higher-level task. See "How to Record a GUI Test" on page 853 or "How to Record a GUI Component" on page 855 for details on:
>
> - Prerequisites
> - Starting a recording session
> - Settings in the Record and Run Settings dialog box or the Applications dialog box
> - Using the Record toolbar

To record in Analog Recording mode:

1. Select the action or scripted component that you want to record.

2. Click the **Record** button or select **Record > Record**.

   If the "Record and Run Settings Dialog Box" or the Applications dialog box opens, set the required settings.

3. In the Record toolbar, select **Analog Recording** from the **Recording Mode** drop down. The Analog Recording Settings dialog box opens. For details, see "Analog Recording Settings Dialog Box " on page 863.

4. Start recording, and perform the operations you want to record in analog recording mode.

You can also use the Record toolbar to manage your recording session, and record steps into different actions or add checkpoints or output value steps, as you would in a normal recording session.

5. When you are finished and want to return to normal recording mode, select **Default** from the **Recording Mode** drop down in the Record toolbar to turn off the option.

How to Record Using Low-Level Recording

Relevant for: GUI tests and scripted GUI components

This task describes how to record a test or scripted component using low-level recording, in which all of your keyboard input and mouse clicks are recorded based on mouse coordinates. When UFT runs the test or scripted component, the cursor retraces the recorded clicks.

Note: This task is part of a higher-level task. See "How to Record a GUI Test" on page 853 or "How to Record a GUI Component" on page 855 for details on:

- Prerequisites
- Starting a recording session
- Settings in the Record and Run Settings dialog box or the Applications dialog box
- Using the Record toolbar

To record in low-level recording mode:

1. Select the action or scripted component that you want to record.

2. Click the **Record** button or select **Record > Record**.

   If the "Record and Run Settings Dialog Box" or the Applications dialog box opens, set the required settings.

3. In the Record toolbar, select **Low-Level Recording** from the **Recording Mode** drop down.

4. Navigate through your application. UFT records each low-level operation you perform and adds it to your test.

   You can also use the Record toolbar to manage your recording session, and record steps into different actions or add checkpoints or output value steps, as you would in a normal recording session.

5. When you are finished and want to return to normal recording mode, select **Default** from the **Recording Mode** drop down in the Record toolbar to turn off the option.

How to Record a Test or Component Using Insight Recording

Relevant for: GUI tests and scripted GUI components

This task describes how to record a test or scripted component using Insight recording, in which UFT recognizes controls based on their appearance, and not their native properties.

When UFT runs the test or component, it recognizes the controls in the application by matching them to the images saved with each of the Insight test objects.

> **Note:** This task is part of a higher-level task. See "How to Record a GUI Test" on page 853 or "How to Record a GUI Component" on page 855 for details on:
>
> - Prerequisites
> - Starting a recording session
> - Settings in the Record and Run Settings dialog box or the Applications dialog box
> - Using the Record toolbar

1. Record operations using the Insight recording mode

a. Select the action or component that you want to record.

b. Click the **Record** button or select **Record > Record**.

If the "Record and Run Settings Dialog Box" or the Applications dialog box opens, set the required settings.

c. In the Record toolbar, select **Insight Recording** from the **Recording Modes** drop down or select **Record > Insight Recording**.

d. Navigate through your application. UFT records each step you perform and adds it to your test.

You can also use the Record toolbar to manage your recording session, and record steps into different actions or add checkpoints or output value steps, as you would in a normal recording session.

> **Note:** There may be a delay after you perform the step in the application and before the step is recorded. You can follow the progress of the recording by checking the number of recorded steps, which is displayed in the Record toolbar's title bar.

e. When you are finished and want to return to normal recording mode, select **Default** from the **Recording Mode** drop down in the Record toolbar to turn off the option.

2. **Fine-tune the recorded test objects - optional**

You may want to make changes in the recorded test objects, to improve the readability and efficiency of your test or component.

In the "Change Test Object Image / Add Insight Test Object Dialog Box", described on page 1233, you can:

- Adjust the borders of the image saved with the test object in the object repository.
- Take a new snapshot to replace the image entirely.
- Specify areas to exclude from the test object image. UFT will ignore these areas when it searches for the image on the screen to identify the object.
- Modify the test object's ClickPoint. This is the location to click in the control when running a test object method on it.

In the object repository you can do any of the following:

- Rename the test object to a name that describes the control it represents. (Recommended)
- Move the test object within the test object hierarchy: If you place it under another test object, then UFT searches for the object in the application only within its parent test object. If you move the Insight test object to be a top-level object, then UFT searches for the object anywhere on the screen.

  For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
- Add a **similarity** identification property to the test object description. For details about this property, see the **InsightObject Identification Properties** section in the *HP UFT Object Model Reference for GUI Testing*.
- Modify the ordinal identifier created for the test object. For details, see "Ordinal Identifiers" on page 1312.
- Define visual relation identifiers for the test object. For details, see "Visual Relation Identifiers" on page 1209.

> **Tips:**
>
> ○ When working in the Editor, you can open the object repository by double-clicking an InsightObject's image. The object repository opens with the InsightObject selected. (If test object images are not displayed in the Editor, you can display them using the

relevant option in the **Tools > Options > GUI Testing** tab > **Insight** pane.)

- When working in the object repository, you can use the **Highlight in Application** option to see which controls UFT identifies using the current test object's description.

3. **Fine-tune the recorded steps - optional**

   In the test or component, you might want to remove unnecessary steps that were recorded or make other adjustments. For example:

   - During Insight recording, when you type in the application, UFT records the **Type** method on a Standard Windows test object, and not on the Insight test object.

     After recording, you can delete this step, and replace it with a **Type** step performed on the relevant Insight test object.

   - During recording, if you click in a control to bring it into focus before typing, UFT records a **Click** step on the relevant Insight test object.

     Similarly, if you press TAB to bring the control into focus before typing, a **Type micTab** step is recorded on the relevant WinObject.

     However, the InsightObject's **Type** method clicks in the control by default before beginning the type operation, rendering the preceding **Click** or **Type** step redundant.

     Therefore, after recording, you can delete the redundant **Click** or **Type** step.

4. **Delete all snapshots from the object repository - optional**

   When you have finished modifying all of the Insight test objects to your satisfaction in the object repository, you can delete all of the snapshots to reduce the amount of disk space used (in the Object Repository window or the Object Repository Manager, **Tools > Delete Insight Snapshots**).

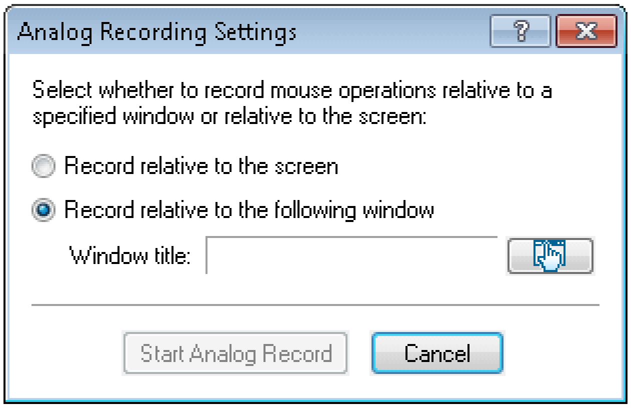
   This does not delete the test object images used for object identification.

Reference

Analog Recording Settings Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to select whether UFT records mouse movements or keyboard input relative to the coordinates of your screen, or relative to the coordinates of the specified window.



Select whether to record mouse operations relative to a specified window or relative to the screen:

○ Record relative to the screen

● Record relative to the following window

Window title:

To access	1. Do one of the following: ▪ Ensure that a GUI test 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. Click the **Record** button to begin a recording session, and do one of the following: ▪ Select **Analog Recording** from the **Recording Mode** drop down in the Record toolbar. ▪ Select **Record > Analog Recording**.

Important information	• You can switch to analog recording mode only while recording. The option is not available while editing. • When you record in analog recording mode **relative to the screen**, the run session will fail if you change the screen resolution or the screen location after recording. • The analog tracking continues to record the movement of the mouse until you drag the mouse to the UFT window (for example, when you want to turn off analog recording or to stop recording). Clicking on the UFT icon in the Windows taskbar is also recorded. This should not affect your run session. The mouse movements and clicks on the UFT screen itself are not recorded. • If you have selected to record in analog recording mode **relative to a window**, any operation performed outside the specified window is not recorded while in analog recording mode.
Relevant tasks	"How to Record Using Analog Recording" on page 858
See also	"How to Record Using Low-Level Recording" on page 859

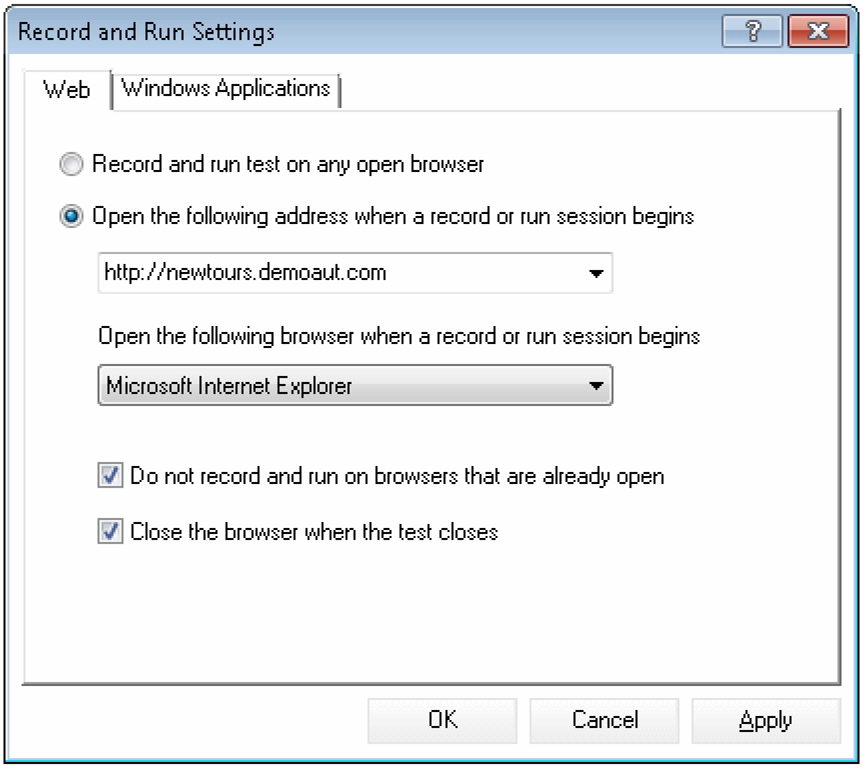
User interface elements are described below:

UI Elements	Description
☝	**Pointing Hand.** Click this button to select the specific area or window of the application to record. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.
Window title	Displays the name of the window to record on in analog recording mode.
Record relative to the screen	UFT records any mouse movement or keyboard input relative to the coordinates of your screen, and disregards any open applications and any applications specified in the Record and Run Settings dialog box. Select this option if you perform your analog operations on objects located within more than one window, or if the window itself may move while you are recording your analog operations.
Record relative to the following window	UFT records any mouse movement or keyboard input relative to the coordinates of the specified window. Select this option if all your operations are performed on objects within the same window and that window does not move during analog recording. This helps ensure that the test or scripted component will run the analog steps in the correct position within the window even if the window's screen location changes after recording.

Record and Run Settings Dialog Box

Relevant for: GUI tests only

This dialog box enables you to set your record and run settings before you begin a record or run session.



○ Record and run test on any open browser

● Open the following address when a record or run session begins

http://newtours.demoaut.com

Open the following browser when a record or run session begins

Microsoft Internet Explorer

☑ Do not record and run on browsers that are already open

☑ Close the browser when the test closes

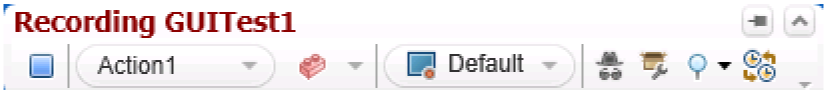
To access	1. Do one of the following: ▪ Ensure that a GUI test or component is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or component node or one of its child nodes. 2. Select **Record > Record And Run Settings**. **Note:** This dialog box opens automatically only before recording a new test, and does not open the next time you record in that test.

Important information	- You can open this dialog box at any time, for example: ▪ If you have already recorded one or more steps in the test, you can modify the settings before you continue recording. ▪ If you want to run the test on a different application than the one you previously used, you can select a different application to test. - The tabs available in the Record and Run Settings dialog box depend on the loaded add-ins. - For details on the tab to use and the options available for the environment you are testing, see the Record and Run Settings for Add-ins Overview in the *HP Unified Functional Testing Add-ins Guide*. - 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 Record and Run Settings dialog box. However, it is important to confirm that the options in 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.
Relevant tasks	"How to Record a GUI Test" on page 853
See also	"Recording Modes" on page 845

Record Toolbar

Relevant for: GUI tests and components

This toolbar enables you to perform, manage, and organize your recording session.



To access	1. Do one of the following: ▪ Ensure that a GUI test 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 **Record > Record**. This toolbar opens automatically at the beginning of a recording session.

Important information	- This toolbar provides the controls for the recording session. From the toolbar, you can choose a type of recording (**default, analog**, **low-level**, **Insight**, or **Standard Windows Recording** (SAP add-in only)) if necessary, add checkpoints to the test or component, add actions to the test, and open the Object Spy dialog box or the Object Repository window. - By default, the toolbar is floating. You can place the toolbar anywhere on the screen or pin it to the top of the screen. - You can minimize the toolbar buttons while recording.
Relevant tasks	- "How to Record a GUI Test" on page 853 - "How to Record Using Analog Recording" on page 858 - "How to Record Using Low-Level Recording" on page 859 - "How to Record a Test or Component Using Insight Recording" on page 860
See also	- "Guidelines for Recording Tests and Components" on page 844 - "Analog Recording" on page 846 - "Low-Level Recording" on page 847 - "Insight Recording" on page 849

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

UI Element	Description
<title bar>	The name of the test or component into which the recorded steps are inserted and the number of steps recorded.
■	**Stop.** Stops the current recording session.
Action1 ▾ **(tests only)**	**<Action List>.** The list of actions in the current test. You can select an action from the drop-down list to assign the recorded steps to different actions.

UI Element	Description
(tests only)	**Call to New Action.** Inserts a call to an action during the recording session. The following options are available: • **Call to New Action** • **Call to Copy of Action** • **Call to Existing Action** • **Call to Existing API Test/Action** For details on the different types of actions, see "Calls to Existing Actions and Copies of Actions" on page 875. **Note:** The dialog box that opens after your choice differ depends on which type of action you insert.
Recording Mode	A drop down list enabling you to switch recording modes. When the drop down is collapsed, it displays the current recording mode. Recording mode choices include: • **Default.** The normal recording mode for UFT recording sessions. If no other recording mode is selected, then recording is done in default mode. • **Analog Recording.** Activates Analog Recording mode. Each time you select this mode, the "Analog Recording Settings Dialog Box " (described on page 863) opens. For details on analog recording, see "Analog Recording" on page 846. • **Low-Level Recording.** Activates Low-Level Recording mode. For details on low-level recording, see "Low-Level Recording" on page 847. • **Insight Recording.** Activates Insight Recording mode. For details on Insight recording, see "Insight Recording" on page 849. • **Standard Windows Recording.** Activates Standard Windows Recording mode. For details, see the section on Standard Windows recording in the SAP GUI for Windows User Guide.
	Object Spy. Opens the Object Spy dialog box. For details, see "Object Spy Dialog Box " on page 1189.
	Object Repository. Opens the Object Repository window. For details, see "Object Repository Document Window" on page 1299.

UI Element	Description
	Insert Checkpoint or Output Value. Inserts a checkpoint or output value to your action or component while recording. For details on checkpoints and output values, see "Checkpoint Types" on page 1397 and "Output Values Overview" on page 1488. For details on how to use checkpoints and output values while recording, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419 or "How to Create or Modify an Output Value Step" on page 1494.
	Synchronization Point. Opens the "Add Synchronization Point Dialog Box" (described on page 982), which enables you to add a wait period to your test for a given step.

Troubleshooting and Limitations - Recording

Relevant for: GUI tests and components

This section describes troubleshooting and limitations for recording tests and components.

- For troubleshooting and limitations for learning objects, and recording and running steps, see "Learning objects, running steps, and recording steps (GUI testing only)" on page 661.

- **Visual relation identifier.** UFT does not record the visual relation identifier property when recording steps. This property can be added only manually from the Object Properties dialog box or the Object Repository Manager or window. For details, see "Visual Relation Identifiers" on page 1209.

- **Start menu.** On Windows Vista, Windows 7, Windows Server 2008, Windows Server 2008 R2, Windows 8, or Windows 2012, if you begin a recording session after installing UFT without restarting your computer, UFT cannot record operations on the Windows **Start** menu or **Quick Launch Panel**.

  Workaround: Restart your computer and start a new recording session.

- **Start menu.** UFT does not record launching Windows Help from the **Start** menu.

- **Start menu.** UFT does not record the selection of **Start** menu items that are customized as menus, such as My Computer, Control Panel, or Recent Documents.

  Workaround: Customize the **Start** menu items as links so that UFT can record operations on them or record the activation of these items another way (and not through the **Start** menu).

- Dragging the Record toolbar to the top of the screen in Windows 7/Windows 8 is not allowed by default.

  Workaround: In **Control Panel > Ease of Access > Ease of Access Center > Make the mouse easier to use** settings, disable the **Prevent windows from being automatically arranged when moved to the edge of the screen** option.

Chapter 33: Actions in GUI Testing

Relevant for: GUI tests only

This chapter includes:

Concepts

Actions in GUI Testing - Overview

Relevant for: GUI tests only

Actions help divide your test into logical units, such as the main sections of a Web site, or specific activities that you perform in your application.

A test is comprised of calls to actions. When you create a new test, it contains a call to a single action. By creating tests that call multiple actions, you can design tests that are more modular and efficient.

An action consists of its own test script, including all of the steps in that action, and any objects in its local object repository and any associated shared object repositories.

Each action is stored together with the test in which you created it. You can insert a call to an action that is stored with the test and, depending on the properties of the action, you may also be able to call an action stored with another test.

When you open a test, you can choose to view the test flow (calls to actions) in the canvas, or you can view and edit the individual actions stored with your test, in the Keyword or Editor.

If you plan tests that will include many steps or lines of script, it is recommended that you use many actions to divide your test steps. Actions should ideally contain no more than a few dozen test steps.

How to Use Actions

Actions enable you to parameterize and iterate over specific elements of a test. They can also make it easier to modify steps in one action when part of your application changes.

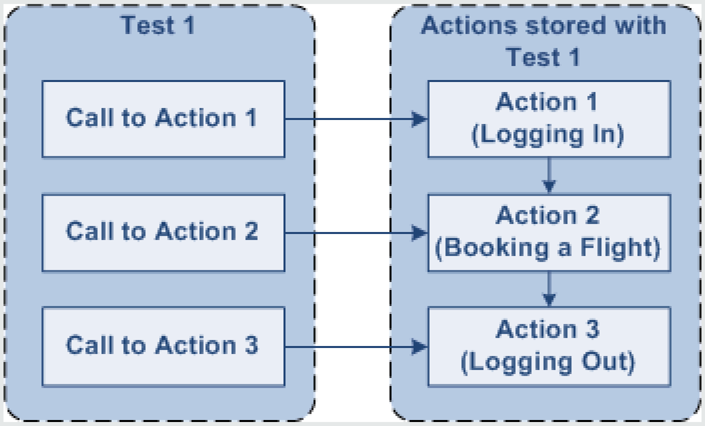
For every action called in your test, UFT creates a corresponding action sheet in the Data pane so that you can enter data table parameters that are specific to that action only. For more information on global and action data sheets, see "Action and Test Iterations Using the Data Pane" on page 874. For information on parameterizing tests, see "Parameterizing Object Values" on page 1526, and "Output Values Overview" on page 1488.

You can also pass information between actions in several ways. You can also specify input parameters for actions, so that steps in an action can use values supplied from other actions in the test. You can also output values from actions to be used in steps in later actions, or to be passed back to the application that ran the test. For more information, see "Action Parameters" on page 877.

Example

Suppose you want to test several features of a flight reservation system. You plan several tests to test various business processes, but each one requires the same login and logout steps. You can create one action that contains the steps required for the login process, another for the logout steps, and other actions for the main steps in your test. After you create the login and logout actions, you can insert those actions into other tests.

If you create a test in which you log into the system, book one flight, and then log out of the system, your test might be structured as shown—one test calling three separate actions:



Action Types

Relevant for: GUI tests only

Each new test automatically includes one blank action. You can add all the steps and make any modifications to your test within this single action.

You can also divide your test into multiple actions by inserting calls to new actions or existing actions. The actions used in the test, and the order in which they are run, are displayed in the canvas. For details, see "The Canvas" on page 209.

When you run a test with multiple actions, the run results are divided by action within each test iteration. This enables you to view the outcome of each action and the detailed results for each action separately. For more information on the Run Results Viewer, see the *HP Run Results Viewer User Guide*.

UFT provides the following types of actions:

Reusable action

An action that can be called multiple times by the local test (the test with which it is stored), as well as by other tests. New actions are reusable by default, and you can mark each action as reusable or non-reusable.

Only reusable actions can be called multiple times from the current test or from another test. Inserting calls to reusable actions makes it easier to maintain your tests, because when an object or procedure in your application changes, the action needs to be updated only once. You must update reusable actions from the original test.

Non-reusable action

An action that can be called only in the local test, and can be called only once. You can store a copy of a non-reusable action with your test and then insert a call to the copy, but you cannot directly insert a call to a non-reusable action saved with another test.

External action

A reusable action stored with another test. External actions are read-only in the calling test, but you can choose to use a local, editable copy of the Data pane information for the external action. For more information, see "Insert Call to New Action Dialog Box (GUI Testing)" on page 913.

Nested action

An action can call another action. This is known as nesting an action, and is described in "How to Nest Actions - Use-Case Scenario" on page 892. Complex tests may have many actions, as well as nested actions, and may share actions with other tests.

Action and Test Iterations Using the Data Pane

Relevant for: GUI tests only

When you use the Data pane to output a value or to add a parameter to your test or action, you can specify whether to store the data in the **Global** data sheet or in the **action** data sheet.

- **Global sheet.** Enables you to define parameters for any action. When you run your test, UFT inserts or outputs a value from or to the current row of the Global data sheet during each global test iteration. This enables you to pass information between actions. For details, see "Global Sheet" on page 226.

- **Action sheet.** Enables you to insert data that applies only to that action. Note that the name of the action sheet is the same as the name of the relevant action. When you run your test, UFT inserts or outputs a value from or to the current row of the current action (local) data sheet during each action iteration. For details, see "Action Sheets" on page 226.

If you create data table parameters or output value steps in your action and select to use the **Current action sheet (local)** option, be sure that the run settings for your action are set correctly in the Run tab of the "Action Call Properties Dialog Box" on page 895.

When you delete an action, the corresponding action sheet is removed from the Data pane, but columns related to this action that are located in the Global sheet are not removed.

For more information, see:

- "Data Pane" on page 221

- "Parameterizing Object Values" on page 1526

- "Output Values Overview" on page 1488

Example

Suppose you want to test how a flight reservation system handles multiple bookings. You may want to parameterize the test to check how your site responds to multiple sets of customer flight itineraries. When you plan your test, you plan the following procedures:

1. The travel agent logs into the flight reservation system.
2. The travel agent books five sets of customer flight itineraries.
3. The travel agent logs out of the flight reservation site.

When you consider these procedures, you realize that it is necessary to parameterize only the second step—the travel agent logs into the flight reservation system only once, at the beginning, and logs out of the system only once, at the end. Therefore, it is not necessary to parameterize the login and logout procedures in your test.

By creating three separate actions within your test—one for logging in, another for booking a flight, and a third for logging out—you can parameterize the second action in your test without parameterizing the others, and without requiring separate actions for each itinerary.

A single test may include both global data table parameters and action (local) data table parameters.

Example

You can create a test in which a travel agent logs into the flight reservation system, books three flights, and logs out; the next travel agent logs into the flight reservation system, books three flights, logs out, and so forth.

To parameterize the book a flight action, you select **Current action sheet (local)** in the "Parameter Options Dialog Box" (described on page 1556) and enter the three flights into the relevant **Action** tab in the Data pane. To parameterize the entire test, you select **Global** in the "Parameter Options Dialog Box" (described on page 1556) and enter the login names and passwords for the different agents into the **Global** tab in the Data pane.

Your entire test runs one time for each row in the Global data sheet. Within each test, each parameterized action is repeated according to the number of rows in its data sheet and the run settings selected in the "Run Tab (Action Call Properties Dialog Box)" (described on page 895).

Note: If your test is stored in ALM, make sure to save your data table parameters in the Global sheet and not in a specific action sheet. For details, see "ALM Integration" on page 707.

Calls to Existing Actions and Copies of Actions

Relevant for: GUI tests only

When you plan a suite of tests or a solution, you may realize that each test requires some identical activities, such as logging in. Rather than inserting the login action three times in three separate tests, and enhancing these actions separately for each test (with checkpoints, parameterization, and programming statements), you can create a single action that logs into the system, and store that action with one test. After you are satisfied with the action you created, you can insert calls to the existing action into other tests.

You can insert calls to existing actions by inserting a call to the original action, which remains read-only in your test, or by inserting a call to a copy of the action, which allows you to customize the action for your test.

You can also call actions dynamically during a run session using the LoadAndRunAction statement. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

Calls to Copies of Actions

When you insert a call to a copy of an action into a test, the original action is copied in its entirety, including checkpoints, parameterization, the corresponding action tab in the Data pane, plus any defined action parameters. If the test you are copying has objects in the local or any associated shared object repository, the copied action's local object repository and associated shared object repository is also copied together with the action.

The action is inserted into the test as an independent action. If the original action was reusable, the new, copied action is also reusable. After the action is copied into your test, you can add to, delete from, or modify the action just as you would with any other non-reusable action. Any changes you make to this action after you insert it affect only this action, and changes you make to the original action do not affect the copied action.

Calls to Existing Actions

You can insert a call to a reusable action that is stored in your current test (local action), or in any other test (external action). Inserting a call to an existing action is similar to linking to it. You can view the steps of the action in the action view, but you cannot modify them. The called action's local object repository (if it has one) is also read-only.

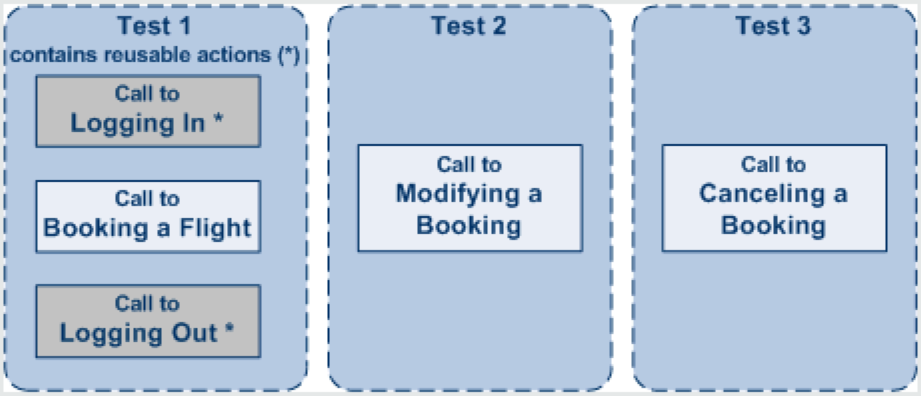
If the called external action has data in the Data pane, however, you can choose whether you want the data from the action's data sheet to be imported as a local, editable copy, or whether you want to use the (read-only) data from the original action. (Columns and data from the called action's global data sheet is always imported into the calling test as a local, editable copy.) For more information, see "External Action Tab (Action Properties Dialog Box)" on page 911.

To modify a called, external action, you must open the test with which the action is stored and make your modifications there. The modifications apply to all tests that call that action. If you chose to use the original action's data when you call an external action, then changes to the original action's data are applied as well.

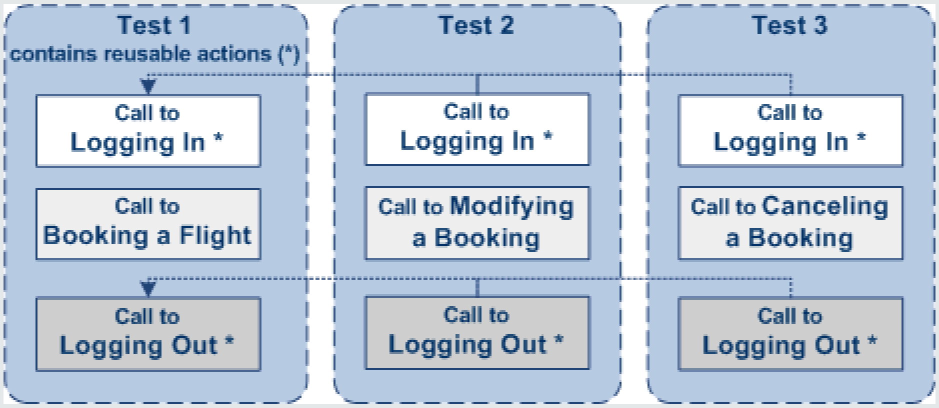
Example

Suppose you want to create the following three tests for the Mercury Tours site—booking a flight, modifying a reservation, and deleting a reservation. While planning your tests, you realize that for each test, you need to log in and log out of the site, giving a total of five actions for all three tests.

You would initially create three tests with five actions. Test 1 would contain two reusable actions (Logging In and Logging Out). These actions can later be called by Test 2 and Test 3.


Test 1
contains reusable actions (*)
Call to Logging In *
Call to Booking a Flight
Call to Logging Out *
Test 2
Call to Modifying a Booking
Test 3
Call to Canceling a Booking

You would then finish creating Test 2 and Test 3 by inserting calls to the reusable actions you created in Test 1.



Action Parameters

Relevant for: GUI tests only

You can specify **input** parameters for an action, so that steps in the action can use values supplied from elsewhere in the test. Input values for an action parameter can be retrieved from any of the following:

- the test (for a top-level action)
- the parameters of the parent action that calls it (for a nested action)
- the output of a previous action call (for a sibling action)

You can specify **output** parameters for an action, so that it can return values for use later in the test. For example, you can output a parameter value to a parent action so that a later nested action can use the output value.

For information on defining action parameters and the values used in action calls, see "Parameters Tab (Action Properties Dialog Box)" on page 903, and "Parameter Values Tab (Action Call Properties Dialog Box)" on page 897.

When to Use Action Parameters

Action parameters enable you to transfer input values from your test to a top-level action, from a parent action to a nested action, or from an action to a sibling action that occurs later in the test. Action parameters also enable you to transfer output values from a step in an action to its parent action, or from a top-level action back to the script or application that ran (called) your test. For example, you can output a value from a step in a nested action and store it in an output action parameter, and then use that value as input in a later step in the calling parent action.

Storing Action Parameters

An action's parameters are stored with the action and are the same for all calls to that action. If you modify an action parameter's name, type, or description, and then view the action properties for a call to that same action in a different part of the test, you will see that the action parameter has changed.

The actual value specified for an input action parameter and the location specified for action output parameter can be different for each call to the action. When you insert a call to a copy of an action, the copy of the action is inserted with the action parameters and action call parameter values that were defined for the action you copied.

Sharing Action Information

Relevant for: GUI tests only

There are several ways to share or pass values from one action to other actions:

- "Output Options Dialog Box " on page 1519. Store values in the output action parameters, and use them in steps performed later in the test, whether in the calling action or sibling actions (for nested actions), or subsequent actions in the test.
- "Sharing Values Using the Global Data Table" below. Store values from one action in the global data table and use these values as data table parameters in other actions.
- "Sharing Values Using Environment Variables" on the next page. Set a value from one action as a user-defined environment variable and then use the environment variable in other actions.
- "Sharing Values Using the Dictionary Object" on the next page. Add values to a Dictionary object in one action and retrieve the values in other actions.

Sharing Values Using the Global Data Table

Relevant for: GUI tests only

You can share a value that is generated in one action with other actions in your test, by storing the value in the global data table. Other actions can then use the value in the Data pane as an input parameter. You can store a value in the Data pane by outputting the value to the global data table, or by using **Data Table**, **Sheet** and **Parameter** objects and methods in the Editor to add or modify a value.

For more information, see:

- "Output Values Overview" on page 1488
- "Parameterizing Object Values" on page 1526
- "Data Pane" on page 221
- The *HP UFT Object Model Reference for GUI Testing*

Example

Suppose you are testing a flight reservation application. When a user logs into the application, the user's full name is displayed on the top of the page. Later, when the user purchases the tickets, the user must enter the name that is listed on his or her credit card.

Suppose your test contains three actions—Login, SelectFlight, and PurchaseTickets and the test is set to run multiple iterations with a different login name for each iteration. In the Login action, you can create a text output value to store the displayed name of the user. In the PurchaseTickets action, you can parameterize the value that is set in the Credit Card Owner edit box using the Data pane column containing the user's full name.

Sharing Values Using Environment Variables

Relevant for: GUI tests only

If you do not need to run multiple iterations of your test, or if you want the value you are sharing to stay constant for all iterations, you can use an internal, user-defined environment variable, which can be accessed by all local actions in your test.

For example, suppose you want to test that your flight reservation application correctly checks the credit card expiration date that the user enters. The application should request a different credit card if the expiration date that was entered is earlier than the scheduled flight departure date. In the SelectFlight action, you can store the value entered in the departure date edit box in an environment variable. In the PurchaseTickets action, you can compare the value of the expiration date edit box with the value stored in your environment variable.

For more information on environment variables, see "Parameterizing Object Values" on page 1526. For information on the **Environment** object, see the *HP UFT Object Model Reference for GUI Testing*.

Sharing Values Using the Dictionary Object

Relevant for: GUI tests only

As an alternative to using environment variables to share values between actions, you can use the Dictionary object. The Dictionary object enables you to assign values to variables that are accessible from all actions called from the test in which the Dictionary object is created, including both local and external actions.

To use the Dictionary object, you must first add a reserved object to the registry (in **HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest Professional\MicTest\ReservedObjects\**) with ProgID = "Scripting.Dictionary".

```
HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest Professional\MicTest\ReservedObjects\GlobalDictionary
```

After you add the reserved Dictionary object to the registry and restart UFT, you can add and remove values to and from the Dictionary in one action, and retrieve the values in another action called from the same test.

For more information on the Dictionary object, see the VBScript Reference documentation (**Help > HP Unified Functional Testing Help > VBScript Reference > Script Runtime**).

Example

Suppose you want to access the departure date set in the SelectFlight action from the PurchaseTickets action. You can add the value of the DepartDate WebEdit object to the dictionary in the SelectFlight action as follows:

```
GlobalDictionary.RemoveAll
```

Then you can retrieve the date from the PurchaseTickets action as follows:

```
Dim CompareDate
CompareDate=GlobalDictionary("DateCheck")
```

Action Syntax in the Editor

Relevant for: GUI tests only

An action call in the Editor can define the action iterations, input parameter values, output parameter storage locations, and an action return values.

This section includes:

- "Calling Actions Using Basic Syntax" on the next page
- "Calling Actions with Parameters" on the next page
- "Storing Action Return Values" on page 882

Calling Actions Using Basic Syntax

Relevant for: GUI tests only

In the Editor, a call to a nested action with no parameters is displayed within the calling action with the following basic syntax:

RunAction *ActionName***,** *IterationQuantity*

Example

To call the **Select Flight** action and run it one iteration:

```
RunAction "Select Flight", oneIteration
```

To call the **Select Flight** action and run it as many iterations as there are rows in the Data pane:

```
RunAction "Select Flight", allIterations
```

To call the **Select Flight** action and run it four iterations (for the first four rows of the Data pane):

```
RunAction "Select Flight", "1 - 4"
```

Calling Actions with Parameters

Relevant for: GUI tests only

If the action you are calling has input and/or output parameters, you can also supply the values for the input parameters and the storage location of the output parameters as arguments of the **RunAction** statement. Input parameters are listed before output parameters.

- For an input parameter, you can specify either a fixed value or you can specify the name of another defined parameter from which the argument should take its value. This can be a data table parameter, an environment parameter, or an action input parameter of the calling action.
- For an output parameter, you can specify either a variable in which you want to store the value or the name of a defined parameter (data table parameter, environment parameter, or an action output parameter of the calling action).

An action call with parameters has the following syntax:

RunAction *ActionName***,** *IterationQuantity***,** *Parameters*

Example

Suppose you call Action2 from Action1, and Action2 has one input and one output parameter defined. The following statement supplies a string value of MyValue for the input parameter and stores the resulting value of the output parameter in a variable called MyVariable.

```
RunAction "Action2", oneIteration, "MyValue", MyVariable
```

The following statement uses the value defined for Action1's Axn1_In input action parameter as the value for the input parameter, and stores the resulting value of the output parameter in Action1's Data pane sheet in a column called Column1_out.

```
RunAction "Action2", oneIteration, Parameter("Axn1_In"),
   DataTable("Column1_out", dtLocalSheet)
```

In the following example, the first statement calls Action2 using its default input parameter value. The second statement uses the value defined for Action2's Axn2_out output action parameter as the value for the call to Action 3's input parameter, and stores the resulting value of the output parameter in Action1's Axn1_out so that the output value is available at the parent action level.

```
RunAction "Action2", oneIteration
RunAction "Action3", oneIteration, Parameter("Action2","Axn2_out"),
    Parameter("Axn1_out")
```

Note that the Action2 output parameter is available for use in the call to Action3, even though no storage location is specified in the call to Action2.

Storing Action Return Values

Relevant for: GUI tests only

If the action called by the **RunAction** statement includes an **ExitAction** statement, the **RunAction** statement can return the value of the **ExitAction**'s *RetVal* argument. Note that this return value is a return value of the action call itself, and is independent of any values returned by specific output parameters of the action call.

To store the return value of an action call, use the syntax:

MyRetVal=**RunAction** (*ActionName*, *IterationQuantity*, *Parameters*)

For more information on the Editor, see "Programming in GUI Testing Documents in the Editor" on page 994. For more information on the **RunAction** statement, see the *HP UFT Object Model Reference for GUI Testing*.

Considerations for Working with Actions

Relevant for: GUI tests only

Inserting Actions

- If you plan to use an identical or virtually identical procedure in more than one test, you should consider inserting a call to an action from another test.

- If you want to make slight modifications to the action in only one test, you should use the **Insert Call to Copy of Action** option to create a copy of the action.

- If you want modifications to affect all tests containing the action, you should use the **Insert Call to Existing Action** option to insert a link to the action from the original test, and make any modifications in the original test.

- If you want modifications to the action to affect all tests containing the action, but you want to edit data in a specific test's data table, use the **Insert Call to Existing Action** option and, in the **External Action** tab of the Action Properties dialog box, select **Use a local, editable copy**.

Storing Actions

- If you expect other users to open your tests and all actions in your tests are stored in the same drive, you should use relative paths for your reusable actions so that other users will be able to open your tests even if they have mapped their network drives differently.

- If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, specify an absolute Quality Center or ALM path. For more information, see "Relative Paths and ALM" on page 758.

Organizing Actions in Your Test

- If your action runs more than one iteration, the action must end at the same point in your application as it started, so that it can run another iteration without interruption. For example, suppose you are testing a sample flight reservation site. If the action starts with a blank flight reservation form, it should conclude with a blank flight reservation form.

- If you expect certain elements of your application to change regularly, it is a good idea to divide the steps related to changeable elements into a separate action so that it will be easy to change the required steps, if necessary, after the application is modified.

Naming Actions

- You may want to rename the actions in your test with descriptive names to help you identify them. It is also a good idea to add detailed action descriptions. This facilitates inserting actions from one test to another. You can rename an action by right-clicking an action in the canvas or the Solution Explorer and selecting **Action Properties**. Make sure you follow the naming conventions for actions, as described in "Troubleshooting - Naming Conventions" on page 2269.

- If a test contains a call to an action stored in another test, and that other test was renamed in ALM, the original test name still appears (in square brackets) in the canvas. The obsolete name in the canvas does not affect UFT's ability to locate and run the action. If it is important to display the correct test name, delete the action call from the test and reinsert it.

- When you rename an action, consider how it will affect your test and any tests that call this action. For example, if you rename an action that is used by another test, the duplication may cause future run sessions may fail because the test cannot locate the specified action.

- If you are working with the Resources and Dependencies model, and the test containing the action you are renaming is stored the Test Plan module in ALM, the internal (default) action name is always displayed in the "Used By Tab (Action Properties Dialog Box)" (described on page 909). This is true even if you rename the action.

Associating Object Repositories

- You can associate as many object repositories as needed with an action, and the same object repository can be associated with different actions as needed. You can also set the default object repositories to be associated with all new actions in a test.

- The order of the object repositories in the list determines the order in which UFT searches for a test object description. If there are test objects in different object repositories with the same name, object class, and parent hierarchy, UFT uses the first one it finds based on the priority order defined in the "Associated Repositories Tab (Action Properties Dialog Box)" (described on page 906). The local object repository is always listed first and cannot be moved down the priority list or deleted.

- You can enter an associated object repository as a relative path. During the run session, UFT searches for the file in the folders listed in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node), in the order in which the folders are listed. For more information, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544.

- You can associate an object repository dynamically during a run session using the RepositoriesCollection statement. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

Action Parameters

For details on using action parameters, see "Considerations for Setting Action Parameters" on page 1530.

Considerations for Removing Action Calls

Relevant for: GUI tests only

- If a test contains a call to an action that you removed, does not exist, or cannot be found, the called action will still appear in the test or calling action, and UFT lists the action in the Errors pane. For more information, see "Errors Pane" on page 364.

- You can remove calls to actions from the following locations:

Location	Description
Solution Explorer	Remove all calls to a specific action. **Note:** If you remove a reusable or non-reusable local action, UFT removes all calls to the action in this test and deletes the action in its entirety. If you remove an external action, UFT removes all calls to the action from the test, but does not affect the source action in any way.
Canvas / Keyword View	Remove specific calls to an action. **Note:** If a test contains multiple calls to a single reusable action, and you remove some—but not all—of the calls, UFT removes the calls to the action in the specified locations, but does not delete the action itself. This means that the action can continue to be called by this test and by other tests, as needed. If you remove all calls to an action, the result is the same as removing the action from the Solution Explorer. For reusable and non-reusable actions, UFT removes all calls to the action in this test and deletes the action in its entirety. For external actions, UFT removes all calls to the action from the test, but does not affect the source action in any way.

Tasks

How to Display and Modify Action-Related Information

Relevant for: GUI tests only

This task includes the following:

- "Display an action in the Keyword View or Editor" below
- "Create an action template" below
- "Display and/or modify action properties" below
- "Display the action call properties" on the next page
- "Display the API test call properties" on page 888
- "Exit an action using programming statements" on page 888

Display an action in the Keyword View or Editor

From the Solution Explorer or the canvas, double-click an action to show only that action in the Keyword View or Editor.

- The **Keyword View** displays the steps of your test in a modular, table format. For details, see "Keyword View" on page 919.
- The **Editor** displays the script for the selected action. For details, see "UFT File Operations" on page 127 and "Programming in GUI Testing Documents in the Editor" on page 994.

You can view and edit the individual steps of an action stored in this test, and view the steps for each selected external action.

Create an action template

1. Create a text file containing the comments, function calls, and other statements that you want to include in your action template. The text file must be in the structure and format used in the Editor.

2. Save the text file as ActionTemplate.mst in your <UFT Installation Folder>\dat folder. All new actions you create contain the script lines from the action template.

   > **Note:** Only the file name ActionTemplate.mst is recognized as an action template.

Display and/or modify action properties

1. Do one of the following:

- Make sure that the action is in focus and select **View > Properties** to display the Properties Pane. For details, see "General Properties Tab (Properties Pane - GUI Testing)" on page 390.

- Right-click an action and select **Action Properties** to open the "Action Properties Dialog Box" (described on page 900). The name of the action and its path are displayed.

2. Use the "Action Properties Dialog Box" (described on page 900) or the "Properties Pane" (described on page 385) to modify action properties. The following tabs are included:

Action Properties dialog box	▪ "General Tab (Action Properties Dialog Box)" on page 900 ▪ "Parameters Tab (Action Properties Dialog Box)" on page 903 ▪ "Associated Repositories Tab (Action Properties Dialog Box)" on page 906 ▪ "Used By Tab (Action Properties Dialog Box)" on page 909 ▪ "External Action Tab (Action Properties Dialog Box)" on page 911
Action Properties pane	▪ "General Properties Tab (Properties Pane - GUI Testing)" on page 390 ▪ "Parameters Tab (Properties Pane - GUI Testing)" on page 394 ▪ "Used By Tab (Properties Pane - GUI Testing)" on page 401

Display the action call properties

Do one of the following:

- In the canvas, right-click an action and select **Action Call Properties** to open the "Action Properties Dialog Box" (described on page 900).

- From the **Parameter presentation** drop-down list, select one of the menu items to display the selected input or output properties adjacent to the action. Select **None** to collapse the properties, and show only the number of input or output properties per action. For details, see "The Canvas" on page 209.

  > **Note:** To view the run values of the action call parameters, see the "Action Call Properties Dialog Box".

Display the API test call properties

Right-click a call to an API test within an action, and select **Edit Call to API Test** to open the "Call to API Test/Action Dialog Box" (described on page 1075).

Exit an action using programming statements

Use one of the following exit action statements:

- **ExitAction.** Exits the current action, regardless of its iteration attributes.
- **ExitActionIteration.** Exits the current iteration of the action.
- **ExitRun.** Exits the test, regardless of its iteration attributes.
- **ExitGlobalIteration.** Exits the current global iteration.

You can view the exit action node in the Run Results tree. If your exit action statement returns a value, the value is displayed in the action, iteration, or test summary, as applicable.

For more information on these functions, see the *HP UFT Object Model Reference for GUI Testing*. For more information on the Run Results, see the *HP Run Results Viewer User Guide*.

How to Manage the Structure of Your GUI Test

Relevant for: GUI tests only

This task includes the following:

- "Insert a call to new action" below
- "Insert a call to existing action or copy of action" on the next page
- "Nest an action within an existing action" on the next page
- "Change the run order of actions" on page 890
- "Remove a call to an action or delete an action" on page 890
- "Delete a call to an API test" on page 891
- "Work with the object repository" on page 891
- "Run or debug a specific action" on page 891

Insert a call to new action

1. Do one of the following:
   - Select **Design > Call to New Action.**
   - In the Solution Explorer, right-click a test and select **Call to New Action**.

- In the canvas, right-click anywhere and select **Call to New Action**.
- In the Record toolbar, click the **Insert Action Call** down arrow and select **Call to New Action** during a recording session.

2. Define the action name and location in the "Action Properties Dialog Box" on page 900.

Insert a call to existing action or copy of action

1. Do one of the following:
   - Select **Design > Call to Copy of Action** or **Design > Call to Existing Action**.
   - In the Solution Explorer, right-click a test and select **Call to Copy of Action** or **Call to Existing Action**.
   - In the canvas, right-click anywhere and select **Call to Copy of Action** or **Call to Existing Action**.
   - In the Record toolbar, click the Insert Action Call down arrow and select **Call to Copy of Action** or **Call to Existing Action** during a recording session.

   For details, see "Calls to Copies of Actions" on page 876 or "Calls to Existing Actions" on page 876.

2. If you are prompted to save your test before continuing, click **Yes** in the dialog box to save your test.

3. Define the action settings in the "Select Action Dialog Box " on page 915.

Nest an action within an existing action

1. In the Keyword View, highlight the step after which you would like to insert the call to the action.

2. Follow the instructions for inserting a call to a new action as described in "Insert Call to New Action Dialog Box (GUI Testing)" on page 913, or for inserting a call to a copy of an action or a call to an existing action as described in "Insert Call to New Action Dialog Box (GUI Testing)" on page 913.

   For a user-case scenario, see "How to Nest Actions - Use-Case Scenario" on page 892.

Change the run order of actions

In the canvas, you can perform any of the following steps to move a top-level action (a direct child of the test) and change the run order of the test accordingly. The action and any nested actions are moved.

- Right-click a top-level action and then select **Move Up** or **Move Down**.
- Select a top-level action and press CTRL+UP arrow or CTRL+DOWN arrow.
- Drag a top-level action up or down to the required location.

> **Note:** You can drag only top-level actions. Selecting the container action automatically includes all of its nested actions. You cannot drag a nested action, and you cannot drag a container action together with only some of its child actions.

Remove a call to an action or delete an action

1. In the Solution Explorer, the canvas, or the Keyword View, do one of the following:
   - Right-click the action you want to remove and select **Delete**.
   - Select the action you want to remove and press the **Delete** key on your keyboard.
   - Select the action you want to remove and select **Edit > Delete**.
2. Click **Yes** in the confirmation message.

> **Note:** If you are deleting a reusable action that is called by another action, and if your test is stored in ALM and is using the resources and dependencies model (described on page 755), a list of the actions that call the action that you are deleting is displayed in the confirmation message box.
> **To copy any or all of the actions from the list to the Windows Clipboard:**
> 1. Select the relevant actions from the list.
> 2. Right-click and select **Copy Selected**, or press CTRL+C on your keyboard.

The following table illustrates what happens when you delete an action:

Action Type	**How deleting the action affects the test:**
Reusable action (action stored in the current test)	▪ If multiple action calls exist in the current test, UFT removes only the call to this action. Additional calls to the action in this test remain unchanged. The corresponding action sheet in the data table remains unchanged. ▪ If this is only call to this action in the current test, UFT deletes the action in its entirety, including its corresponding action sheet in the Data pane. **Caution:** Be careful when deleting a local reusable action. If the action is called by other tests, deleting the action may cause the other tests to fail.
Non-reusable action (action stored in the current test)	Deletes the action in its entirety, including its corresponding action sheet in the Data pane.
External action (action stored in a different test)	Removes the call to the action from the current test without affecting the action in the source test. The original action remains stored with the test in which it was created.

Delete a call to an API test

Right-click a call to an API test and select **Delete** to remove it from the action.

Work with the object repository

Right-click an action and select **Object Repository** to open the Object Repository window, which displays a tree containing all objects in the current test. If there is a shared object repository associated with the action, the shared object repository opens. If there is no shared object repository associated with the action, the local object repository opens. For details, see "Shared Object Repositories" on page 1285.

Run or debug a specific action

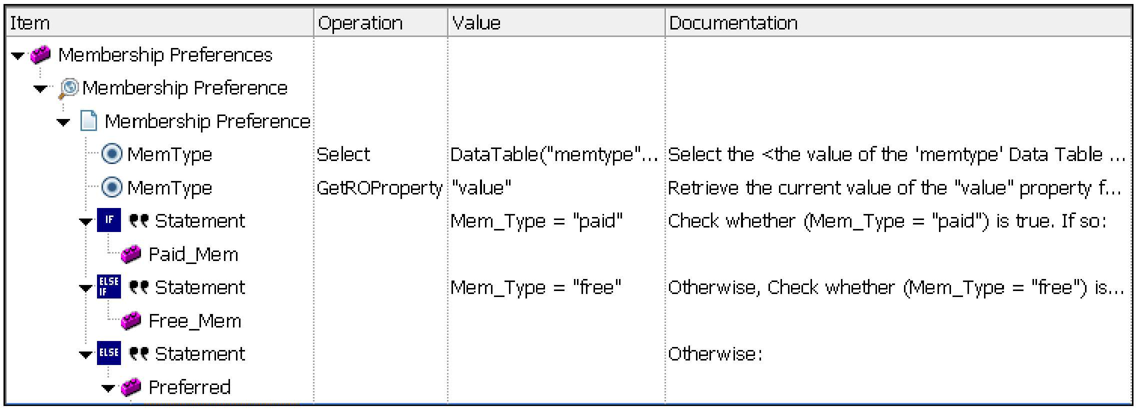
For details, see "How to Debug Your Test, Component, Function Library, or User Code File" on page 683.

How to Nest Actions - Use-Case Scenario

Relevant for: GUI tests only

Suppose you parameterized a step in which a user selects one of three membership types as part of a registration process. When the user selects a membership type, the page that opens depends on the membership type selected in the previous page. You can create a separate action for each type of membership, and then use **If** statements to determine the membership type selected in a particular iteration of the test, and run the appropriate action for that selection.

For more information on inserting conditional statements, see "Conditional Statements" on page 970.



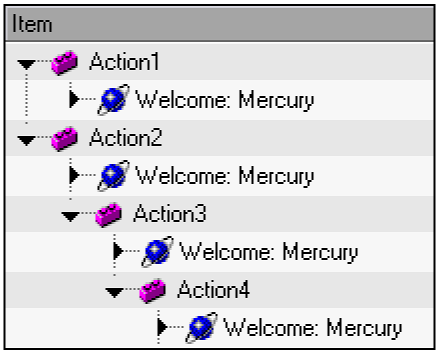
Item	Operation	Value	Documentation
Membership Preferences			
Membership Preference			
Membership Preference			
MemType	Select	DataTable("memtype"...	Select the <the value of the 'memtype' Data Table ...
MemType	GetROProperty	"value"	Retrieve the current value of the "value" property f...
Statement		Mem_Type = "paid"	Check whether (Mem_Type = "paid") is true. If so:
Paid_Mem			
Statement		Mem_Type = "free"	Otherwise, Check whether (Mem_Type = "free") is...
Free_Mem			
Statement			Otherwise:
Preferred			

```
Browser("Membership Preference").Page("Membership Preference").WebRadioGroup("MemTyp
e").Select DataTable("memtype", dtGlobalSheet)
Mem_Type=Browser("Membership Preference").Page("Membership Preference").WebRadioGro
up("MemType").GetROProperty ("value")
If Mem_Type="paid" Then
    RunAction "Paid_Mem", oneIteration
ElseIf  Mem_Type = "free"  Then
    RunAction "Free_Mem", oneIteration
Else
    RunAction "Preferred", oneIteration
End If
```

How to Use Action Parameters - Use-Case Scenario

Relevant for: GUI tests only

Suppose you want to take a value from the external application that runs (calls) your test and use it in an action within your test. If your test has an **Action4** that is nested within **Action3**, and **Action3** is further nested within **Action2**, you would need to pass the input test parameter from the external application, through **Action2** and **Action3**, to the required step in **Action4**.


Item
Action1
Welcome: Mercury
Action2
Welcome: Mercury
Action3
Welcome: Mercury
Action4
Welcome: Mercury

You would do this as follows:

1. Define the input test parameter (**File > Settings > Parameters** node) with the value that you want to use later in the test.

2. Define an input action parameter for Action2 (right-click an action and select **Action Properties > Parameters** tab) with the same value type as the input test parameter.

3. Parameterize the input action parameter value (right-click an action and select **Action Call Properties > Parameter Values** tab) using the input test parameter value you specified above.

4. Define an input action parameter for Action3 (right-click an action and select **Action Properties > Parameters** tab) with the same value type as the input test parameter.

5. Parameterize the input action parameter value.

   - Right-click an action and select **Action Call Properties > Parameter Values** tab and select the input action parameter value you specified for Action2.

   - Use the **Parameter** utility object to specify the action parameter as the *Parameters* argument for the **RunAction** statement in the Editor. For more information, see "Calling Actions with Parameters" on page 881.

6. Define an input action parameter for Action4 (right-click an action and select **Action Properties > Parameters** tab) with the same value type as the input test parameter.

7. Parameterize the input action parameter value.

   - Right-click an action in the canvas or Keyword View and select **Action Call Properties > Parameter Values** tab and select the input action parameter value you specified for Action3.

   - Use the **Parameter** utility object to specify the action parameter as the *Parameters* argument for the **RunAction** statement in the Editor. For more information, see "Calling Actions with Parameters" on page 881.

8. Parameterize the value in the required step in Action4.

   - Click the parameterization icon and specify the parameter in the "Value Configuration Options Dialog Box" (described on page 1579) using the input action parameter you specified for Action 4.

   - Use the **Parameter** utility object in the Editor to specify the value to use for the step. For more information, see "Using Action Parameters in Steps in the Editor" on page 1529.

> **Note:** You can also modify many of the action properties from the Properties pane. For details, see "Properties Pane User Interface (GUI Testing)" on page 387.

Reference

Action Call Properties Dialog Box

Relevant for: GUI tests only

This dialog box controls the way the action behaves in a specific call to the action. It enables you to specify how many times UFT should run the called action (according to the number of rows in the Data pane), as well as the initial value for any input action parameters and the location in which you want to store the values of any output action parameters.

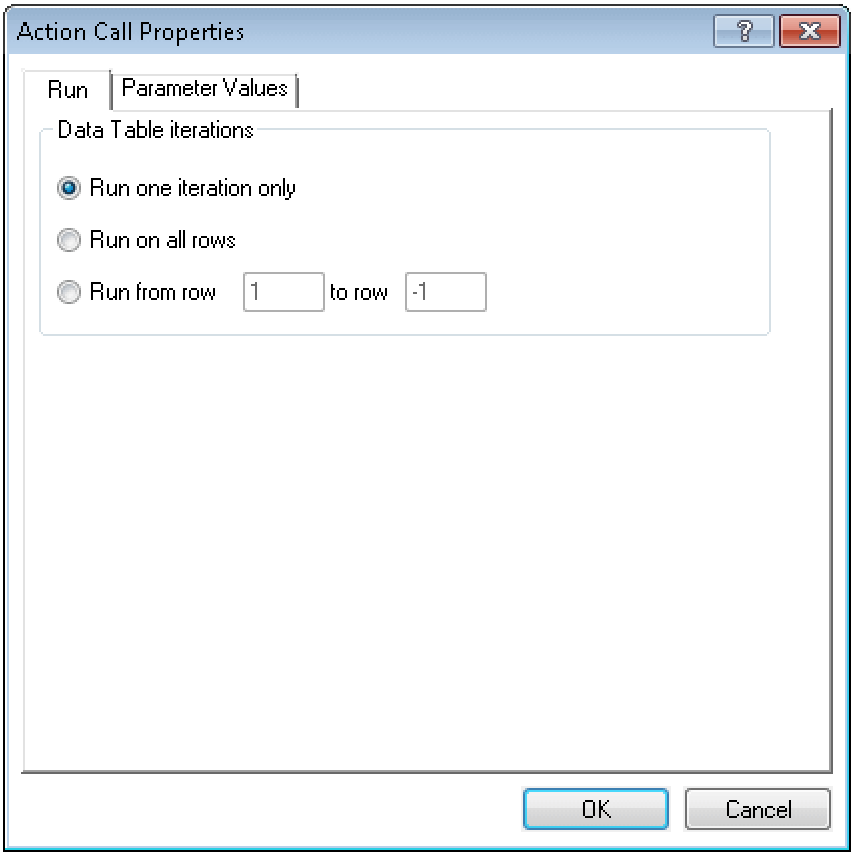
To access	1. Select a GUI test tab or action tab. 2. If you are working with an action, click the **Keyword View** toggle button or select **View > Keyword View /Editor**. 3. Right-click an action in the canvas or an action node in the Keyword View and select **Action Call Properties**.
Important information	- This dialog box enables you to set options that apply only to a specific action call. - You can also define action calls and action call parameters in the Editor. For more information, see "Action Syntax in the Editor" on page 880
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886

The Action Call Properties dialog box contains the following tabs:

Run Tab (Action Call Properties Dialog Box)

Relevant for: GUI tests only

This tab enables you to instruct UFT to run only one iteration on the called action, to run iterations on all rows in the Data pane, or to run iterations only for a certain row range in the Data pane.



● Run one iteration only

○ Run on all rows

○ Run from row 1 to row -1

To access	In the "Action Call Properties Dialog Box" (described on page 895), select the **Run** tab.
Important information	- If you run multiple iterations on an action, the action must begin and end at the same point in the application, so that the application is in the proper location and state to run the next iteration of the action. - This tab applies to individual action calls and refers to the rows in the action's data sheet. You can set the Run properties for an entire test (setting iterations for rows on the Global data sheet) from the Run pane in the Test Settings dialog box. For more information, see "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886
See also	"Parameter Values Tab (Action Call Properties Dialog Box)" on the next page

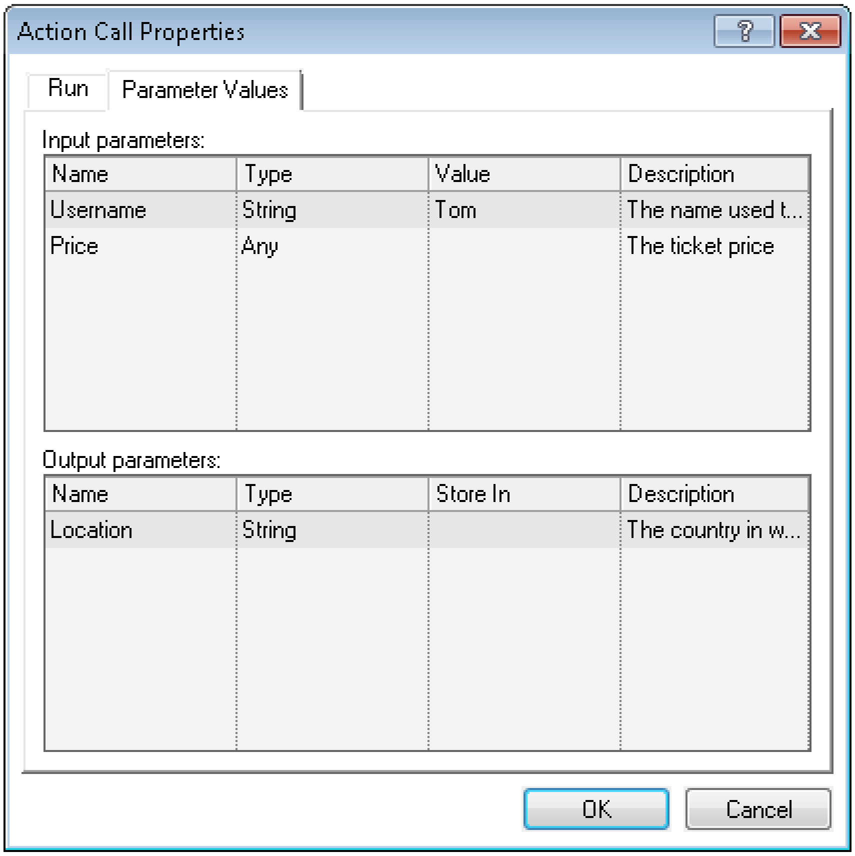
User interface elements are described below:

Option	Description
Run one iteration only	Runs the called action only once, using the first row in the action's data sheet.
Run on all rows	Runs the called action with the number of iterations according to the number of rows in the action's data table.
Run from row __ to row __	Runs the called action with the number of iterations according to the specified row range.

Parameter Values Tab (Action Call Properties Dialog Box)

Relevant for: GUI tests only

This tab enables you to specify the values of input action parameters used by the called action and to specify the locations in which you want to store output action parameter values. You can also parameterize the value used for a particular input action parameter using any available parameter type.



Input parameters:

Name	Type	Value	Description
Username	String	Tom	The name used t...
Price	Any		The ticket price

Output parameters:

Name	Type	Store In	Description
Location	String		The country in w...

To access	In the "Action Call Properties Dialog Box" (described on page 895), select the **Parameter Values** tab.
Important information	- Specifying input and output parameter values in action calls is optional. - If you do not set a value for an input action parameter, the default value that is specified in the Action Properties dialog box is used. - If you do not define a storage location for an output parameter value, the calling action still has access to the output parameter data generated by the actions it calls. However, specifying a storage location can make your action call statements more readable. - You can also modify parameters from the Properties pane. - This tab enables you to link action parameters with test parameters.
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886
See also	- "Run Tab (Action Call Properties Dialog Box)" on page 895 - "Parameters Tab (Properties Pane - GUI Testing)" on page 394

User interface elements are described below:

UI Elements	Description
Name	The name for the parameter. (Action parameter names are case sensitive.)

UI Elements	Description
Type	The value type for the parameter. The following options are available: - **String.** A character string enclosed within a pair of quotation marks, for example, "New York". If you enter a value and do not include the quotation marks, UFT adds them automatically when the value is inserted in the script during the test run. The default value is an empty string. - **Boolean.** A true or false value. If you select this value type, you can click in the **Default Value** column and click the arrow to select a **True** or **False** value. The default value is **True**. - **Date.** A date string, for example, 3/2/2005. If you select this value type, you can click in the **Default Value** column and click the arrow to open a calendar from which you can select a date. The default value is today's date. - **Number.** Any number. The default value is 0. - **Password.** An encrypted password value. If you select this value type, the password characters are masked when you enter the password in the **Default Value** field. In the action, however, the value appears encrypted. The default value is an empty string, which also appears as an encrypted value in the actual action. - **Any.** A variant value type, which accepts any of the above value types. If you select this value type, you must specify the value in the format that is required in the location where you intend to use the value. For example, if you intend to use the value later as a string, you must enclose it in quotation marks. When you specify a value of **Any** type, UFT checks whether it is a number. If the value is not a number, UFT automatically encloses it in quotation marks. If you are editing an existing value, UFT automatically encloses it in quotation marks if the previous value had quotation marks. The default value is an empty string.
Value (Input Parameters)	The default value for the parameter. You can leave the default value provided for the parameter value type. The default value is required so that you can run the action without receiving parameter values from elsewhere in the test.
Store In (Output Parameters)	Opens the "Storage Location Options Dialog Box" (described on page 991), enabling you to specify how and where to store a return value for the output parameter.
Description	The description of the parameter, for example, the purpose of the parameter in the action. This description is displayed together with the name of the parameter in any dialog box in which you can choose an action parameter, including the Output Options, Parameter Options, and Value Configuration Options dialog boxes.

Action Properties Dialog Box

Relevant for: GUI tests only

This dialog box enables you to define options for the stored action. These settings apply each time the action is called. You can modify an action name, add or modify an action description, and set an action as reusable or non-reusable.

To access	1. Select a GUI test tab or action tab. 2. If you are working with an action, click the **Keyword View** toggle button or select **View > Keyword View /Editor**. 3. Right-click an action in the canvas or an action node in the Keyword View and select **Action Properties**.
Important information	You can also define actions and action parameters in the Editor or the Properties pane.
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886
See also	• "Action Syntax in the Editor" on page 880 • "General Properties Tab (Properties Pane - GUI Testing)" on page 390

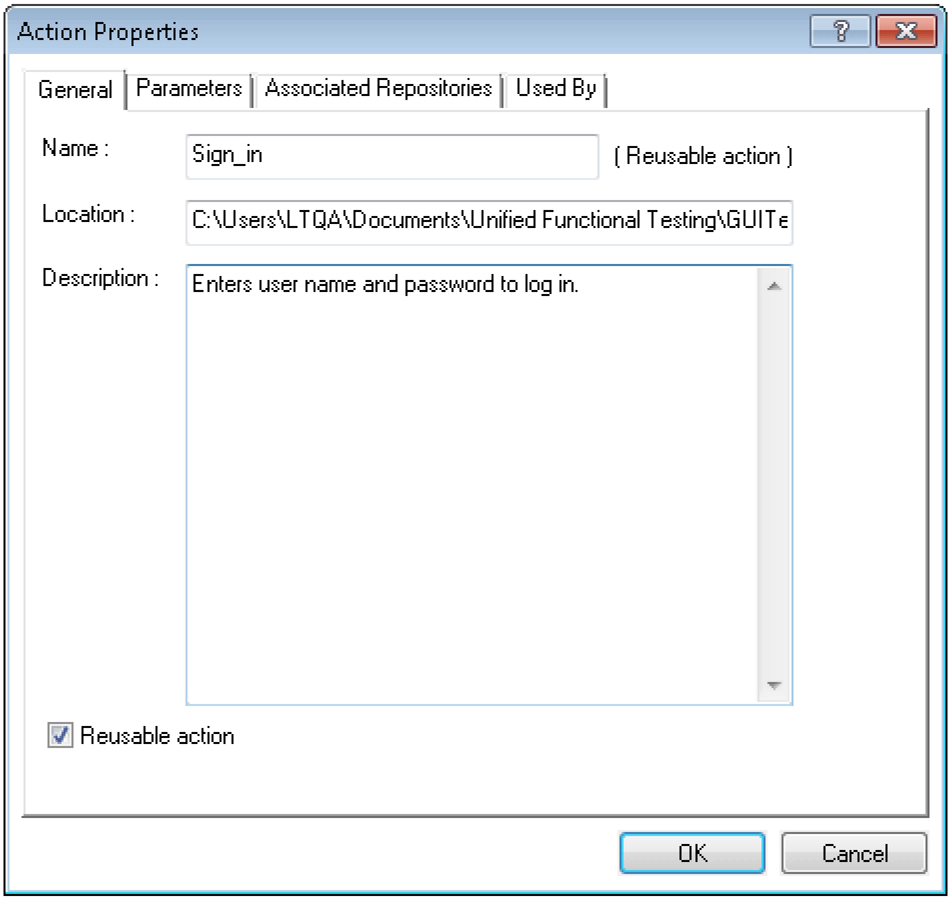
This dialog box contains the following tabs:

- General tab (described on page 900). Enables you to modify the name of an action, add or edit an action's description, or change the reusability status of the action.
- Parameters tab (described on page 903). Enables you to define input and output parameters to be used by the action.
- Associated Repositories tab (described on page 906). Enables you to specify the object repositories that are associated with the action.
- Used By tab (described on page 909). Enables you to view a list of the tests and actions that contain calls to this particular action. Available only if your tests are stored in ALM and are using the resources and dependencies model.
- External Action tab"External Action Tab (Action Properties Dialog Box)". Enables you to set the data table definitions. Available only when viewing properties for external actions. When this tab is displayed, the other tabs are read-only.

General Tab (Action Properties Dialog Box)

Relevant for: GUI tests only

This tab enables you to modify the name of an action, add or edit an action's description, or change the reusability status of the action.



Name : Sign_in (Reusable action)

Location : C:\Users\LTQA\Documents\Unified Functional Testing\GUITe

Description : Enters user name and password to log in.

☑ Reusable action

To access	In the "Action Properties Dialog Box" (described on page 900), select the **General** tab.
Important information	• The name of the action and its path are displayed in the tab. If it was defined with a relative path in UFT, then the path is displayed as <test name>\<action name>. • If the action is called more than once within the test flow or if the action is called by a reusable action, the **Reusable action** option is read-only. If you want to make the action non-reusable, remove the additional calls to the action from the test. • You can view details of a reusable action by double-clicking the action. For more information on the test flow and action views, see "How to Display and Modify Action-Related Information" on page 886. • You can also modify many action parameters from the Parameters pane.
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886

See also	• "Parameters Tab (Action Properties Dialog Box)" on the next page • "Associated Repositories Tab (Action Properties Dialog Box)" on page 906 • "Used By Tab (Action Properties Dialog Box)" on page 909 • "External Action Tab (Action Properties Dialog Box)" on page 911 • **Properties pane:** "General Properties Tab (Properties Pane - GUI Testing)" on page 390

User interface elements are described below:

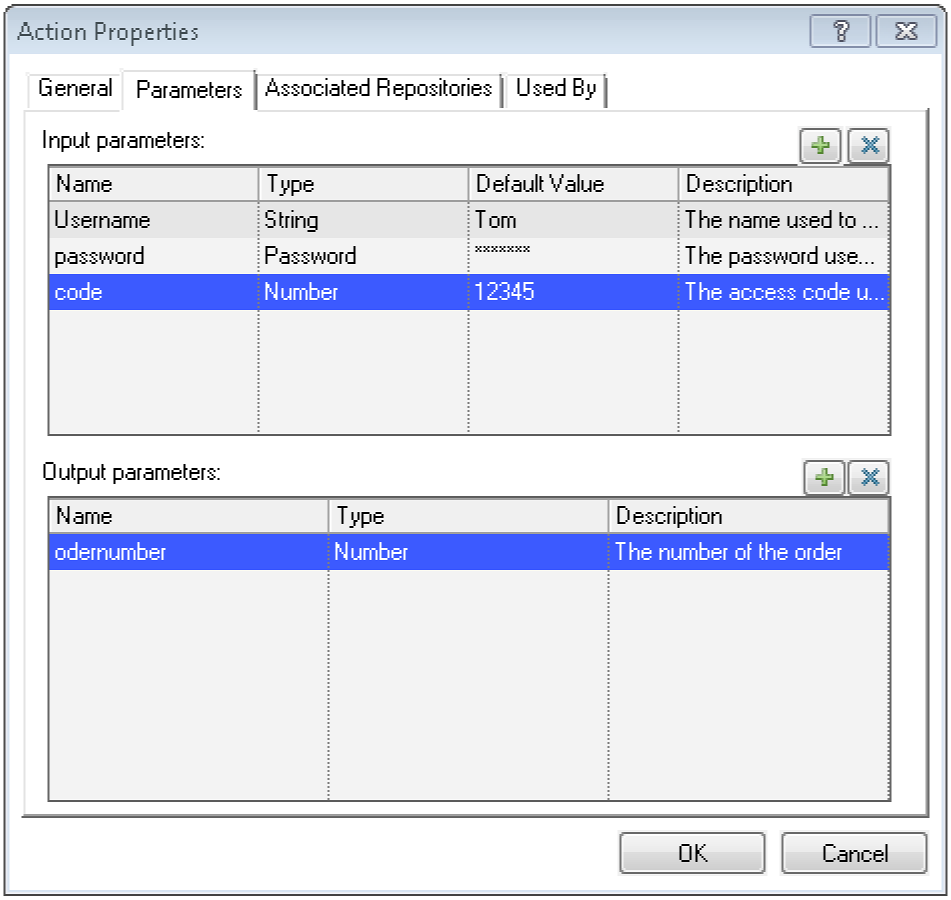
UI Elements	Description
Name	The name of the action. By default, the action name is the internal name provided by UFT, such as Action 1. This number is incremented by 1 for each new action that is added to the test. If the action is a reusable action or an external action, then **Reusable action** or **ExternalAction** is displayed next to the action name. You can rename the action, as needed. When renaming an action, keep the following points in mind: • Consider how the new name will affect your test and any tests that call this action. For example, if you rename an action that is used by another test, future run sessions may fail because the test cannot locate the specified action. • If you are working with the Resources and Dependencies model, and the test containing the action you are renaming is stored the Test Plan module in ALM, the internal (default) action name is always displayed in the "Used By Tab (Action Properties Dialog Box)" described on page 909. This is true even if you rename the action. For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269.
Location	The folder or ALM path where the action is stored.

UI Elements	Description
Description	You can insert comments about the action. An action description helps you and other testers know what a specific action does without reviewing all the steps in the action. The description is also displayed in the description area of the "Select Action Dialog Box " (described on page 915). This enables you and other testers to determine which action you want to call or copy from another test without having to open it. For more information on inserting copies and calls to actions, see "How to Nest Actions - Use-Case Scenario" on page 892. **Note:** You can also add a description when inserting a call to a new action. For more information, see "Insert Call to New Action Dialog Box (GUI Testing)" on page 913.
Reusable action	Indicates whether the action is a reusable action. By default, this check box is selected. A reusable action can be called multiple times within a test and can be called from other tests. Non-reusable actions can be copied and inserted as independent actions, but cannot be inserted as calls to the original action. If the steps of the action were expanded, they collapse after changing a non-reusable action to a reusable action. You can view the steps of the reusable action by selecting the action name in the canvas. **Note:** If an action stored in this test is called by other tests, deleting the action in this test may cause other tests to fail.

Parameters Tab (Action Properties Dialog Box)

Relevant for: GUI tests only

This tab enables you to define input and output parameters for use in the selected action's iterations.



Input parameters:

Name	Type	Default Value	Description
Username	String	Tom	The name used to ...
password	Password	*******	The password use...
code	Number	12345	The access code u...

Output parameters:

Name	Type	Description
odernumber	Number	The number of the order

To access	In the "Action Properties Dialog Box" (described on page 900), select the **Parameters** tab.
Important information	• When you delete an action parameter, make sure that you also delete or modify any steps that use the action parameter. • You can also modify action parameters from the Properties pane.
Relevant tasks	• "How to Display and Modify Action-Related Information" on page 886
See also	• "General Tab (Action Properties Dialog Box)" on page 900 • "Associated Repositories Tab (Action Properties Dialog Box)" on page 906 • "Used By Tab (Action Properties Dialog Box)" on page 909 • "External Action Tab (Action Properties Dialog Box)" on page 911 • "Parameters Tab (Properties Pane - GUI Testing)" on page 394

User interface elements are described below:

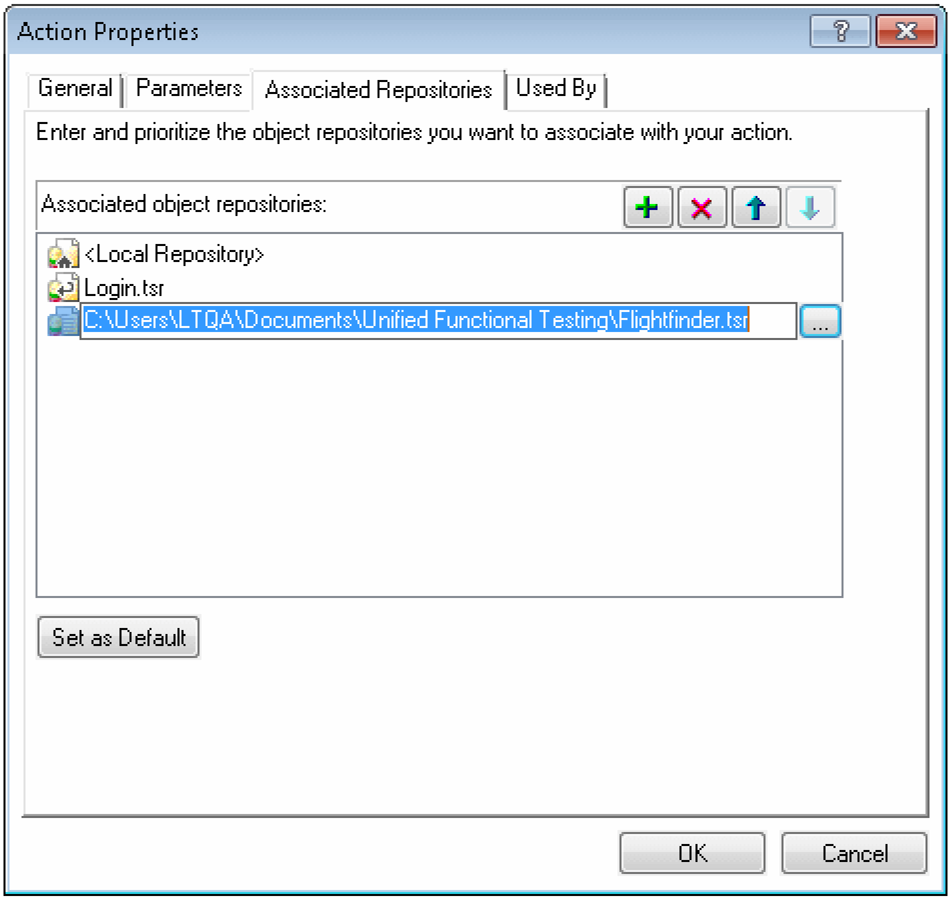
UI Elements	Description
✚	**Add.** Adds a new parameter to the list.
✕	**Delete.** Removes the selected parameter from the list.
Name	The name for the parameter. For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269.
Type	The value type for the parameter. The following options are available: • **String.** A character string enclosed within a pair of quotation marks, for example, "New York". If you enter a value and do not include the quotation marks, UFT adds them automatically when the value is inserted in the script during the test run. The default value is an empty string. • **Boolean.** A true or false value. If you select this value type, you can click in the **Default Value** column and click the arrow to select a **True** or **False** value. The default value is **True**. • **Date.** A date string, for example, 3/2/2005. If you select this value type, you can click in the **Default Value** column and click the arrow to open a calendar from which you can select a date. The default value is today's date. • **Number.** Any number. The default value is 0. • **Password.** An encrypted password value. If you select this value type, the password characters are masked when you enter the password in the **Default Value** field. In the action, however, the value appears encrypted. The default value is an empty string, which also appears as an encrypted value in the actual action. • **Any.** A variant value type, which accepts any of the above value types. If you select this value type, you must specify the value in the format that is required in the location where you intend to use the value. For example, if you intend to use the value later as a string, you must enclose it in quotation marks. When you specify a value of **Any** type, UFT checks whether it is a number. If the value is not a number, UFT automatically encloses it in quotation marks. If you are editing an existing value, UFT automatically encloses it in quotation marks if the previous value had quotation marks. The default value is an empty string.
Default Value (Input Parameters)	The default value for the parameter. You can leave the default value provided for the parameter value type. The default value is required so that you can run the action without receiving parameter values from elsewhere in the test.

UI Elements	Description
Description	The description of the parameter, for example, the purpose of the parameter in the action. This description is displayed together with the name of the parameter in any dialog box in which you can choose an action parameter, including the Output Options, Parameter Options, and Value Configuration Options dialog boxes.

Associated Repositories Tab (Action Properties Dialog Box)

Relevant for: GUI tests only

This tab enables you to associate object repositories with an action, and manage the association order and default settings.



Enter and prioritize the object repositories you want to associate with your action.

Associated object repositories:

<Local Repository>
Login.tsr
C:\Users\LTQA\Documents\Unified Functional Testing\Flightfinder.tsr

To access	In the "Action Properties Dialog Box" (described on page 900), select the **Associated Repositories** tab.

Important information	- You can also associate shared object repositories with multiple actions simultaneously, using the "Associate Repositories Dialog Box" (described on page 1266). - If UFT is not connected to an ALM project, all associated object repositories that are stored in your ALM project are listed as missing in the Errors pane. For details, see "Errors Pane User Interface" on page 374. - If an object repository cannot be found, a warning message displays when you click this tab. UFT also adds a question mark to the missing object repository icon to the left of the missing object repository in the **Associated object repositories** list. For details, see "How to Manage Errors in the Errors Pane" on page 369.
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886
See also	- "General Tab (Action Properties Dialog Box)" on page 900 - "Parameters Tab (Action Properties Dialog Box)" on page 903 - "Used By Tab (Action Properties Dialog Box)" on page 909 - "External Action Tab (Action Properties Dialog Box)" on page 911

User interface elements are described below:

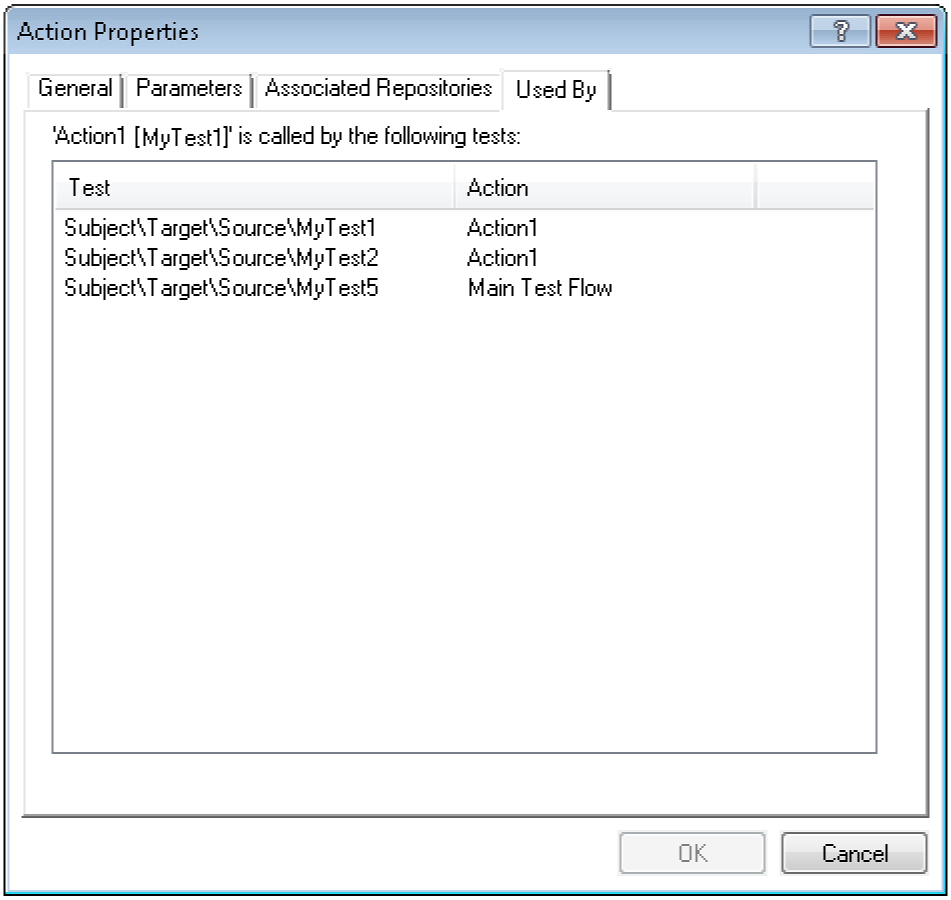
UI Elements	Description
Associated object repositories	The list of all associated object repositories. **Note:** If you want other users or HP products to be able to run an action on other computers, and the action's associated object repositories are stored in the file system, you can set the file path as a relative path (click the path once to highlight it, and then click it again to enter edit mode). Any users who want to run this action should then specify the drive letter and folder in which UFT should search for the relative path in the Folders pane of the Options dialog box (**Tools > Options> GUI Testing** tab **> Folders** node). For more information, see "Folders Pane (Options Dialog Box > GUI Testing Tab) " on page 544, and "Relative Paths in UFT for GUI Testing" on page 130.

UI Elements	Description
+	Associates an object repository with the action. You can enter the absolute or relative path and filename of the object repository, or use the browse button to locate the required file. You can associate object repositories that are saved in your file system or in an ALM project. **Note:** To use the Resources and Dependencies model with Quality Center 10.00 or ALM, specify an absolute Quality Center or ALM path. For more information, see "Relative Paths and ALM" on page 758. **Tips for adding an ALM path:** • When connected to ALM, click this button. UFT adds [QualityCenter], and displays a browse button so that you can locate the ALM path. • When not connected to ALM, hold the Shift key and click this button. UFT adds [ALM], and you enter the path. You can also type the entire ALM path manually. If you do, you must add a space after [ALM]. For example: [ALM] Subject\ObjectRepositories\flight.tsr.
✕	Removes an associated object repository from the list.
↑	Assigns a higher priority to the selected object repository.
↓	Assigns a lower priority to the selected object repository.
Set as Default	Sets the current list of object repositories as the default list to be associated with all new actions in this test. **Note:** The **Set as Default** option is enabled only when: • One or more shared object repositories are associated with any local action in the test. • The list of object repositories associated with this action is different from the list associated with other local actions in this test.

Used By Tab (Action Properties Dialog Box)

Relevant for: GUI tests only

This tab enables you to view a list of the tests and actions that contain calls to this particular action. This tab is available only if your tests are stored in ALM and are using the resources and dependencies model.



'Action1 [MyTest1]' is called by the following tests:

Test	Action
Subject\Target\Source\MyTest1	Action1
Subject\Target\Source\MyTest2	Action1
Subject\Target\Source\MyTest5	Main Test Flow

To access	In the "Action Properties Dialog Box" on page 900 (described on page 900), select the **Used By** tab.
Important information	• This list is the same list that is displayed in the Dependencies tab of the Test Plan module in ALM. For more information, see "Resources and Dependencies Model" on page 754. • Calls to external actions that are stored in Quality Center 10.00 or ALM via a relative path are not listed in this pane because they are not using the resources and dependencies model. • You can also view this list from the "Used By Tab (Properties Pane - GUI Testing)" (described on page 401).

Relevant tasks	"How to Display and Modify Action-Related Information" on page 886
See also	- "General Tab (Action Properties Dialog Box)" on page 900 - "Parameters Tab (Action Properties Dialog Box)" on page 903 - "Associated Repositories Tab (Action Properties Dialog Box)" on page 906 - "External Action Tab (Action Properties Dialog Box)" on the next page - **Properties pane:**"Used By Tab (Properties Pane - GUI Testing)" on page 401

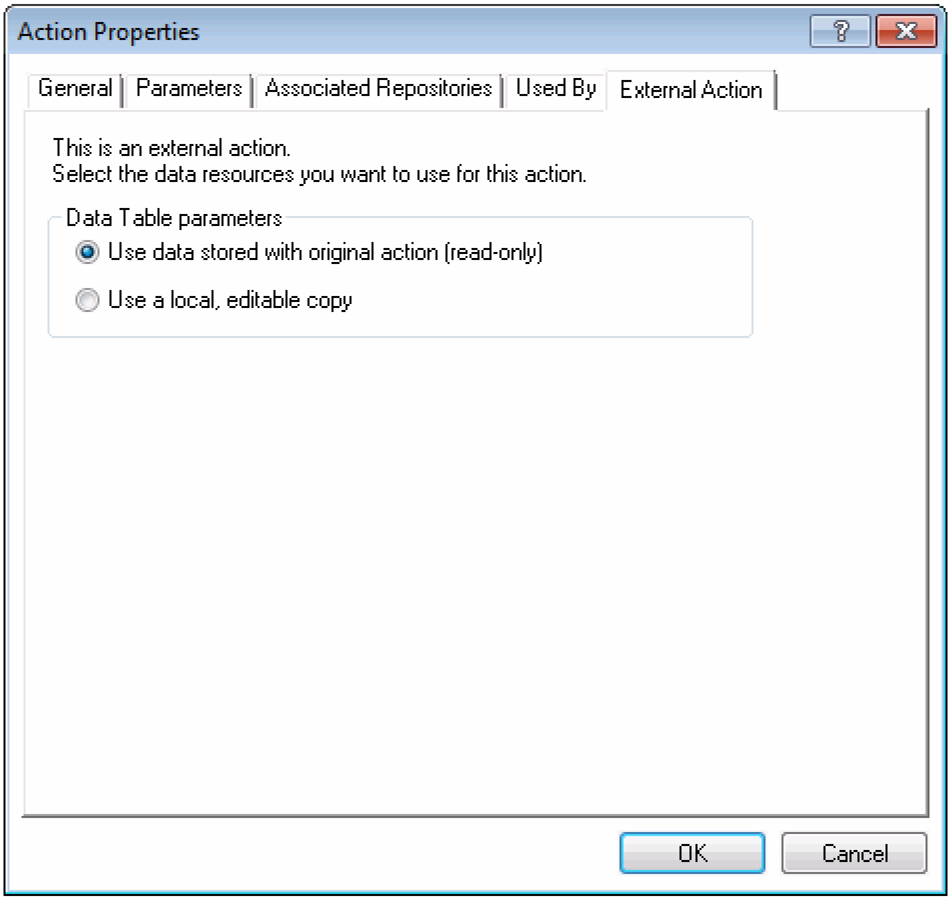
User interface elements are described below:

UI Elements	Description
Test	The ALM path of the test containing a call to this action.
Action	The name of the action containing a call to this action. If the called action is the top-level action in the test from which it is called, **Main Test Flow** is displayed instead.

External Action Tab (Action Properties Dialog Box)

Relevant for: GUI tests only

This tab enables you to specify the data source for the selected action. This tab is available only when viewing properties for external actions.



This is an external action.
Select the data resources you want to use for this action.

Data Table parameters

● Use data stored with original action (read-only)

○ Use a local, editable copy

To access	In the "Action Properties Dialog Box" (described on page 900), select the **External Action** tab.
Important information	- When this tab is displayed, the other tabs are read-only. - When you insert a call to an external action, the action is inserted in read-only format, and the **Record** button is disabled. If you want to record, you first need to insert a call to a reusable or non-reusable action into your test, or select a step from a reusable or non-reusable action that already exists in your test.
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886

See also	- "General Tab (Action Properties Dialog Box)" on page 900 - "Parameters Tab (Action Properties Dialog Box)" on page 903 - "Associated Repositories Tab (Action Properties Dialog Box)" on page 906 - "Used By Tab (Action Properties Dialog Box)" on page 909

User interface elements are described below:

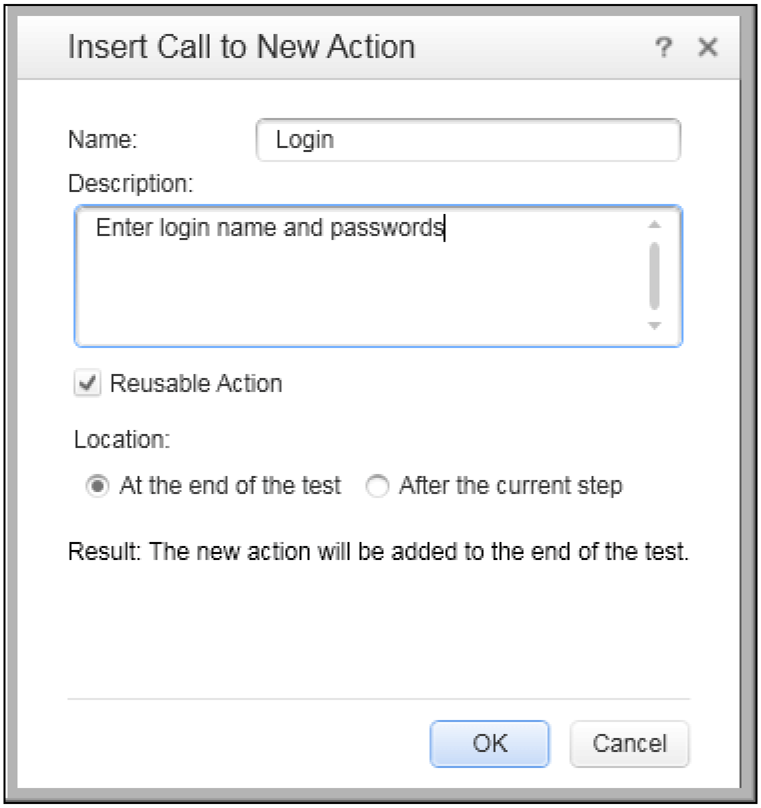
UI Elements	Description
Data Table parameters	Indicates where to store the data in the action's data table: - **Use data stored with the original action (read-only).** Uses the original action's data. If you select this option, the data is read-only when viewed from the calling test, and all changes to the original action's data sheet apply when the action runs in the calling test. - **Use a local, editable copy**. Uses an editable copy of the data in the test's data table. If you select this option, a copy of the called action's data sheet is added to the calling test's data table and is independent of the original action. Changes to the original action's data sheet do not affect the calling test even if you insert another call to this action after the action's data sheet is modified. If the called action has parameterized steps that rely on new information in the original action's data sheet, enter the relevant column names and required data to the action sheet in the calling test manually. **Note:** When you call an external action, the global data sheet columns and data from the called action's test are always imported as a local, editable copy in the calling test's global data sheet. Changes to the original action's global data sheet do not affect the calling test even if you insert another call to this action after the called action's global data sheet is modified. If the called action has parameterized steps that rely on new information in the global data sheet, enter the relevant column names and required data to the calling test's global data sheet manually.

Insert Call to New Action Dialog Box (GUI Testing)

Relevant for: GUI tests only

For API testing, see "Insert Call to New Action Dialog Box (API Tests) " on page 1796.

This dialog box enables you to insert a call to a new action.



Insert Call to New Action

Name: Login

Description:

Enter login name and passwords

☑ Reusable Action

Location:

● At the end of the test ○ After the current step

Result: The new action will be added to the end of the test.

To access	1. Do one of the following: - Ensure that a GUI test or action is in focus in the document pane. - In the Solution Explorer, select a GUI test or action node. 2. Do one of the following: - Select **Design > Call to New Action**. - Click the **Insert Call to New Action** toolbar button . - Click the **Call to New Action** button in the Record toolbar during a recording session and select **Call to New Action**.

Important information	• You can call the new action from your test flow as a top-level action, or you can call the new action from within another action in your test as a sub-action (or nested action). For more information, see "How to Nest Actions - Use-Case Scenario" on page 892. • When you click **OK**, a new action is stored with your test and the call to it is added at the bottom of the test or after the current step. • You can move your action call to another location at a parallel (sibling) level within your test by dragging it to the desired location. For more information on moving actions, see "How to Move an Action or Component Step" on page 935.
Relevant tasks	"How to Display and Modify Action-Related Information" on page 886

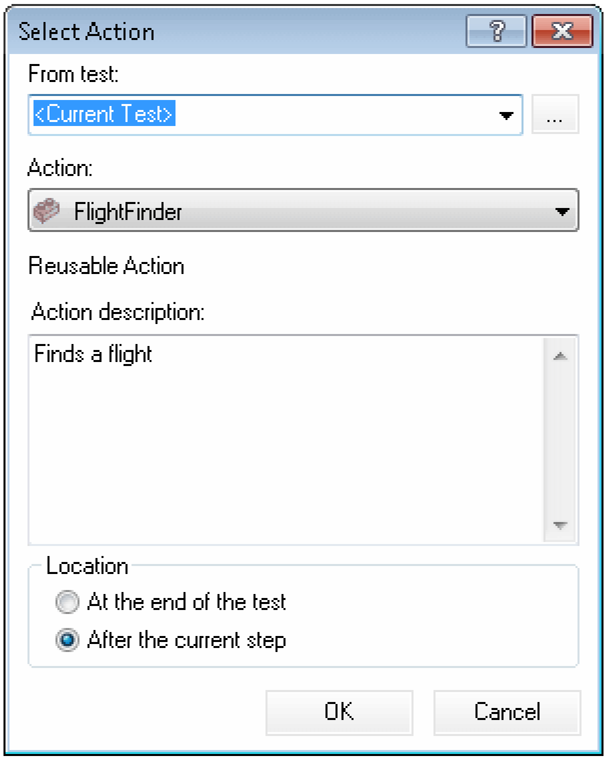
User interface elements are described below:

UI Elements	Description
Name	Enables you to type a new action name or accept the default name. If you rename the action, see "Troubleshooting - Naming Conventions" on page 2269.
Description	The description of the action. You can also add an action description at a later time using the Properties pane. **Note:** Action descriptions are displayed in the Select Action dialog box when you add a call to an existing action. The description helps you to choose the action you want to call. For more information, see "General Tab (Action Properties Dialog Box)" on page 900.
Reusable Action	When selected, enables you to call the action from other tests or multiple times from within this test (default). You can set or modify this setting at a later time using the Properties pane.
Location	The location to insert the new action. The following options are available: • **At the end of the test.** Creates a call from the test flow to a top-level action. • **After the current step**. Inserts the call to the action from within the current action (nests the action). **Note:** If the currently selected step is a reusable action from another test, the new action is added automatically to the end of the test (the location options are disabled).

Select Action Dialog Box

Relevant for: GUI tests only

This dialog box enables you to select actions to use in your test as copies or as external actions.



From test: <Current Test>

Action: FlightFinder

Reusable Action

Action description: Finds a flight

Location
○ At the end of the test
● After the current step

To access	1. Do one of the following: - Ensure that a GUI test or action is in focus in the document pane. - In the Solution Explorer, select a GUI test or action node. 2. Select one of the following: - **Design > Call to Copy of Action/Call to Existing Action** - Click the **Insert Call to New Action** down arrow in the toolbar and select **Call to Copy of Action/Call to Existing Action** from the drop-down list. - Click the **Call to New Action** down arrow in the Record toolbar during a recording session and select **Call to Copy of Action/Call to Existing Action** from the drop-down list.

Important information	• When inserting a call to copy of action, you can view the location of the original action in the General tab of the Action Properties dialog box. • When inserting a call to existing action, you can create an additional call to any reusable or external action in your test by pressing CTRL while you drag and drop the action to another location at a parallel (sibling) level within your test.
Related tasks	"How to Display and Modify Action-Related Information" on page 886
See also	"Keyword View Overview" on page 921

User interface elements are described below:

UI Elements	Description
From test	The name of the test containing the action you want to use. You can also enter an ALM folder or a relative path. If you enter a relative path, UFT searches for the test in the folders listed in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node). For more information, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544 and "Relative Paths in UFT for GUI Testing" on page 130. If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, specify an absolute Quality Center / ALM path. For more information, see "Relative Paths and ALM" on page 758.
Action	All local actions (actions that are stored with the test you selected). When you select an action, its type (**Non-reusable Action** or **Reusable Action**) and description, if one exists, are displayed. This helps you identify the action you want to copy. For more information on action descriptions see "General Tab (Action Properties Dialog Box)" on page 900. **Note:** When inserting an existing action, external actions that the test calls are also displayed in the list. If the action you want to call is already called from within the selected test, you can select it from the list of actions. This creates another call to the original action.
Edit new action properties (Call to Copy of Action only)	When selected, the "Action Properties Dialog Box" (described on page 900) is displayed when you click **OK**. You can then modify the action properties. **Tip:** If you do not select this option, you can modify the action's properties later by right-clicking the action icon in the Keyword View and selecting **Action Properties**.

UI Elements	Description
Location	The location to insert the action. The following options are available: - **At the end of the test** - **After the current step** If the currently selected step is a reusable action from another test, the call to the copy of the action is added automatically to the end of the test, and the **After the current step** option is disabled.

Troubleshooting and Limitations - Actions

Relevant for: GUI tests only

This section describes troubleshooting and limitations for working with actions.

- If you make a copy of an existing test (either in the file system or in ALM), then you cannot insert a call to the same action from both of these tests into the same test.

  Workaround: Instead of creating a copy of the test, use **Save As** to create a duplicate of the test.

- You cannot create a call to a new action called **Global**, since Global is the name reserved for the Global sheet in the Data pane. If you create an action called Global, you will not be able to select the local or global data sheet when parameterizing a identification property.

- Calling an external action from a function library using the **RunAction** statement results in a run-time error.

  Workaround: Call the action using the **LoadAndRunAction Statement**. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

- You cannot add a new action as a nested action to an external action.

  Workaround: Open the external action and add the call to the nested action directly.

Chapter 34: Keyword View

Relevant for: GUI tests and components

This chapter includes:

Concepts

Keyword View Overview

Relevant for: GUI actions and components

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

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

Additional Details for Business Components

The Keyword View that you see in UFT is the same as the Automation tab in ALM.

The Keyword View for components differs from the Keyword View for actions in that it provides component-specific options that are specially designed to facilitate the creation of business components. For example, unlike action steps (test objects and operations) that are displayed hierarchically in the Keyword View, component steps are displayed at the same hierarchical level, even if they are child objects of the previous step or operations to be performed.

To work with components in the Keyword View, UFT must be connected to an ALM project with BPT support.

Types of Action and Component Steps

Relevant for: GUI actions and components

You can insert the following types of steps using the relevant options from the **Insert** or shortcut menus.

- A **standard** step. For details, see "How to Add a Standard Step to Your Test or Component" on page 925.

- A **checkpoint** step. For details, see "Checkpoints Overview" on page 1396.

- An **output value** step.

  For actions, see: "Output Values Overview" on page 1488

  For components, see: "Output Value for Your Component Step - Output Column" on the next page

- **Comments** in steps to separate parts of an action or a component and to add details about a specific part.

  For details, see:

  - For actions: "Insert Comment Dialog Box" on page 962
  - For components: "Comments in the Keyword View" below

For actions and scripted components, you can also insert:

- A step that uses **programming logic** to:

  - send information to the run results
  - put a comment line in your test
  - synchronize your test with your application
  - measure a transaction in your test

  For details, see "Generated Programming Operations" on page 968.

- Steps containing **conditional statements** and **loop statements**. For details, see "Conditional and Loop Statements in the Keyword View" on the next page and "Standard Steps After a Conditional or Loop Block" on page 924.

Comments in the Keyword View

Relevant for: GUI actions and components

A **Comment** is a free text entry. You can use comments to improve readability and help make an action or component easier to update. For example, you may want to add a comment to describe what is being tested. Alternatively, you may want to use comments to plan your action or component steps before your application is ready to be tested. Later, you can use your plan to verify that every item that needs to be tested is included in the component steps.

The 💬 icon indicates a comment in the Keyword View. In actions, comments are displayed in the **Comment** cell of a step. This column is hidden by default, so you may need to show it to see comments. In components, comments are displayed as a separate step. Comments are always visible as long as one or more Keyword View columns are displayed.

For details on selecting columns to display, see "Columns Tab (Keyword View Options Dialog Box) " on page 954.

Output Value for Your Component Step - Output Column

Relevant for: GUI components

You define the output type and settings for the output value in the **Output** cell. These determine where the output value is stored and how it is used during the component run session. When the output value step is reached, UFT retrieves each value selected for output and stores it in the specified location for use later in the run session.

When you create a new output value step, UFT assigns a default definition to each value selected for output. When you output a value for a step in a business component:

- If at least one output component parameter is defined in the component, the default output type is **Component parameter** and the default output name is the first output parameter displayed in the Parameters pane of the Business Component Settings dialog box.
- If no output component parameter is defined in the component, the default output type is **Local parameter** and the default name is **p_Local**.

You modify the output parameter, as required. If you select a local parameter, you can modify it directly in the "Output Options Dialog Box " (described on page 1519). If you select a component parameter, the details for the output parameter are read-only. You can modify the parameter details in the Parameters pane of the Business Component Settings dialog box. For details, see "Parameters Tab (Properties Pane - GUI Testing)" on page 394.

If, after you specify an output value, you choose not to save the output value, you can cancel it. For details, see "How to Navigate in the Keyword View and Other Tips" on page 937.

Conditional and Loop Statements in the Keyword View

Relevant for: GUI actions and scripted GUI components

Using **conditional statements**, you can incorporate decision making into your tests. Using **loop statements**, you can run a group of steps repeatedly, either while or until a condition is true. You can also use loop statements to repeat a group of steps a specific number of times.

The following conditional statements are available in the Keyword View:

Icon	Type	Description
IF	**If...Then**	Performs a statement or a series of statements based on specified conditions. If a condition is not fulfilled, the next **Elseif** condition or **Else** statement is examined.
ELSE IF	**ElseIf...Then**	
ELSE	**Else**	

The following loop statements are available in the Keyword View:

Icon	Statement	Description
↻	**While...Wend**	Performs a series of statements as long as a specified condition is True.
FOR	**For...Next**	Uses a counter to perform a group of statements a specified number of times.
↻	**Do...While**	Performs a series of statements indefinitely, as long as a specified condition is True.
↻	**Do...Until**	Performs a series of statements indefinitely, until a specified condition becomes True.

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

> **Note:** For more details on loop statements, see the VBScript documentation (select **Help > HP Unified Functional Testing Help > VBScript Reference**).

For task details, see:

- "How to Insert Conditional Statements from the Keyword View" on page 930
- "How to Insert Loop Statements In the Keyword View" on page 933.
- "How to Add a Standard Step After a Conditional or Loop Block" on page 934

Standard Steps After a Conditional or Loop Block

Relevant for: GUI actions and scripted GUI components

After you add a conditional or loop statement to your test, all steps that you add or record are automatically inserted within the conditional or loop statement block.

When you finish adding steps to the block, you can add a step outside of the block, at a sibling level to the conditional or loop statement step, as described in "How to Add a Standard Step After a Conditional or Loop Block" on page 934. For details on conditional and loop statements, see "Conditional and Loop Statements in the Keyword View" on the previous page and "Generated Programming Operations" on page 968.

Tasks

How to Add a Standard Step to Your Test or Component

Relevant for: GUI actions and components

This task includes the following steps:

- "Insert a new step" below
- "Define the step " below

1. Insert a new step

Do any of the following:

- Click anywhere in the Keyword View (below the existing steps, if any) to add a step at the end of the action or component. If no steps are defined yet, this adds the first step to the action or component.
- Right-click an existing step and select **Insert New Step** from the shortcut menu.
- Drag and drop a test object from the Toolbox pane to an action displayed in either the Keyword View or the Editor.

  For actions: A new step is added to the Keyword View, either as a sibling step or a sub-step of an existing step, according to the test object hierarchy, as described in "UFT Test Object Hierarchy" on page 1174.

  For components: A new step is added to the Keyword View below the selected step.

2. Define the step

Click in the cell for the part of the step you want to modify and specify its contents, as described below. Each cell in the step row represents a different part of the step. For each step, you can define the following:

- **Item** (described on page 944 for actions, and on page 945 for components). For task details, see "How to Select an Item for Your Step " on the next page.
- **Operation** (described on page 946). For task details, see "How to Select an Operation for Your Step " on page 927.
- **Value** (described on page 947). For task details, see "How to Define Values for Your Step Arguments " on page 928.
- **Assignment (actions only)** (described on page 948). For task details, see "How to Navigate in the Keyword View and Other Tips" on page 937.
- **Comment** (described on page 948). For task details, see "How to Navigate in the Keyword

View and Other Tips" on page 937.

- **Output (components only)** (described on page 948). For additional details, see "Output Value for Your Component Step - Output Column" on page 923.

> **Tip:** You can use the standard editing commands (**Cut**, **Copy**, **Paste**, and **Delete**) in the **Edit** menu or in the shortcut menu to make it easier to define or modify your steps. You can also drag and drop steps to move them to a different location within your action or component. For details, see "Keyboard Shortcuts in the Keyword View" on page 953.

How to Select an Item for Your Step

Relevant for: GUI actions and components

This task describes how to select an item for your step and includes the following steps:

- "Select an item from the displayed list" below
- "Drag and drop an item from the Toolbox pane" below
- "Select a test object from an associated object repository or from your application" below

Do one of the following:

Select an item from the displayed list

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

> **For components:** To specify a function for the step (instead of selecting a test object), select **Operation**. Then specify an operation for the step as described in "How to Select an Operation for Your Step " on the next page.

Drag and drop an item from the Toolbox pane

Drag and drop a test object from the Toolbox pane into your action or component. A step with the default operation for that test object is inserted.

For more details, see "Toolbox Pane User Interface (GUI Testing)" on page 510.

Select a test object from an associated object repository or from your application

1. Click in the **Item** cell to activate it.

2. Click the down arrow and select **Object from repository** (for tests) or **Select another object** (for components) to open the "Select Test Object Dialog Box" (described on page 958).

   For tests: The test objects available in the list are the sibling and child test objects of the previous step's test object. You can select whether you want the operation for the step to be a test object operation or a run-time object operation. If you select a run-time object, an **Object** statement is added to the Keyword View.

How to Select an Operation for Your Step

Relevant for: GUI actions and components

This task describes how to specify the operation to be performed on the item listed in the **Item** column.

This task includes the following steps:

- "Prerequisite" below
- "Select an operation" below

1. Prerequisite

Select an item in the **Item** cell. For details, see "How to Select an Item for Your Step " on the previous page.

For components:

- If you selected a test object in the **Item** column, all selected operations for the test object item (as defined in the Keywords pane in the application area) are automatically displayed in the **Operation** column. The **Operation** list for that test object can include out-of-the-box operations and any user-defined functions that were registered to that specific test object type.
- If you selected **Operation** in the **Item** column, all functions defined in associated function libraries are displayed, alphabetically. (You manage the function libraries for components in the Function Libraries pane of the Application Area. For details, see "Function Libraries Pane (Application Area)" on page 2102.)

2. Select an operation

a. Click in the **Operation** cell to activate it.

b. Click the down arrow and select the operation to be performed on the item.

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

For components: The operation can be either a standard operation or a user-defined function that is enabled for use in the Keywords pane of the application area. For details on user-defined functions, see "User-Defined Functions and Function Libraries" on page 1029 and "Keywords Pane (Application Area)" on page 2107.

Tip: When you position the cursor over an operation in the list, a tooltip describes what this operation performs. For user-defined functions, the tooltip is taken from the description that you provided in the associated function library. For details, see "Add documentation details to the function - optional" on page 1053.

How to Define Values for Your Step Arguments

Relevant for: GUI actions and components

This task describes how to specify the arguments for the operation to be performed on the item listed in the **Item** column.

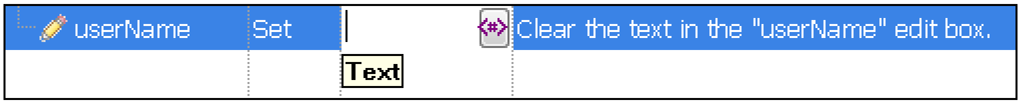
This task includes the following steps:

- "Define or modify a value " below
- "Add multi-line arguments" on the next page
- "Parameterize the value for an argument " on page 930
- "Encode a password" on page 930

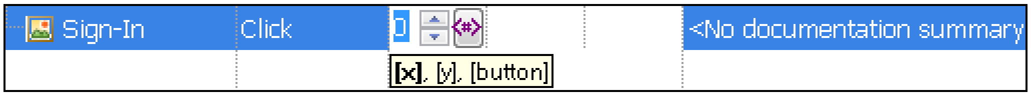
Define or modify a value

1. Click in the **Value** cell to activate it.

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



Some methods may display multiple partitions. For example:



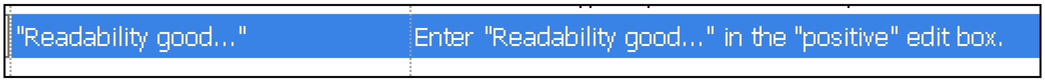
Sign-In	Click	0 [x], [y], [button]		<No documentation summary

You can enter **constant** or **parameterized** values. For details on parameterizing values, see "Parameterize the value for an argument " on the next page.

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

Add multi-line arguments

In a **Value** cell, press SHIFT+ENTER to add line breaks to your argument value. After you enter a multi-line argument value, the value is automatically converted to a string, and only the first line of the argument is displayed, followed by an ellipsis (...). This format for multi-line argument values is also displayed in the Documentation column of the Keyword View.



"Readability good..."	Enter "Readability good..." in the "positive" edit box.

Note:

- You can select the cell to display the entire argument value to be used in the step.
- The argument value is used during the run session exactly as it appears in the step. For example, if you enter quotation marks as part of the argument value, they are included in the argument value used during the run session.
- Multi-line values are automatically interpreted as strings, so you do not need to add quotation marks for this purpose.

Parameterize the value for an argument

Click the <#> button in the required **Value** cell. The Value Configuration Options dialog box opens. For user interface details, see:

- **For actions:**"Value Configuration Options Dialog Box" on page 1579
- **For components:**
  - **Local input parameters** in the Business Component Keyword View: "Value Configuration Options Dialog Box" on page 1579
  - **Local output parameters** in the Output Options dialog box: "Output Options Dialog Box " on page 1519

> **Note:** You cannot delete local parameters, but you can cancel the input or output to them. For details on parameterizing a **component** value, see "Parameters Tab (Properties Pane - GUI Testing)" on page 394.

Encode a password

You can encode passwords for use in method arguments and data table cells. For details on how to encode passwords, see:

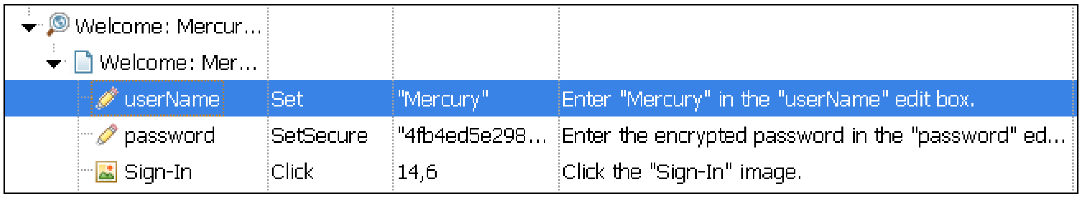
- **For actions:**"Password Encoder Tool" on page 965.
- **For components stored in ALM.** The *HP Business Process Testing User Guide*. Do not encode passwords in UFT. When business component tests run from ALM, all values are treated as strings. Therefore, if you encode a password in UFT using the "Password Encoder Tool" (described on page 965), the resulting string, for example, 4b8a4a999d0d0e2c9b6cce8bb8e3, will be used instead of the password it represents.
- **For components stored in UFT.**"Password Encoder Tool" on page 965. You can use the password encoder to insert passwords in UFT. (When your components are later upgraded to ALM, the upgrade process automatically converts existing encoded strings to a format recognized by ALM.)

How to Insert Conditional Statements from the Keyword View

Relevant for: GUI tests and scripted GUI components

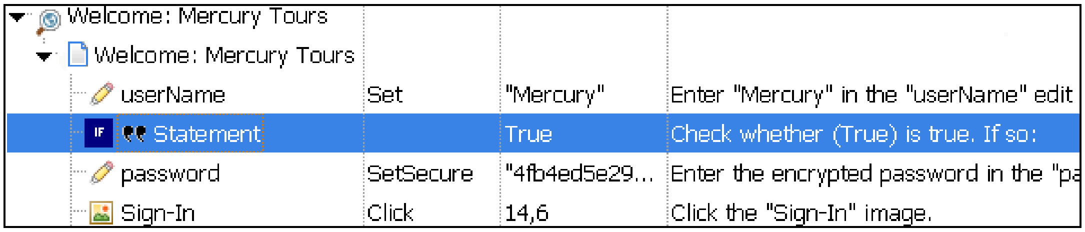
This task describes how to insert a conditional **If...Then...Else** statement from the Keyword View.

1. In the Keyword View, select the step that you want the conditional statement to follow. The following example shows the userName row selected:



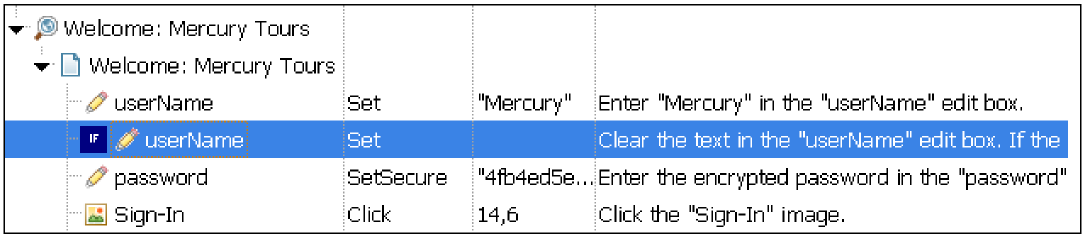
Welcome: Mercur...			
Welcome: Mer...			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
password	SetSecure	"4fb4ed5e298...	Enter the encrypted password in the "password" ed...
Sign-In	Click	14,6	Click the "Sign-In" image.

2. Select **Edit > Code Snippet** and select **If...Then**. The new statement is added to the Keyword View below the selected step. For example:



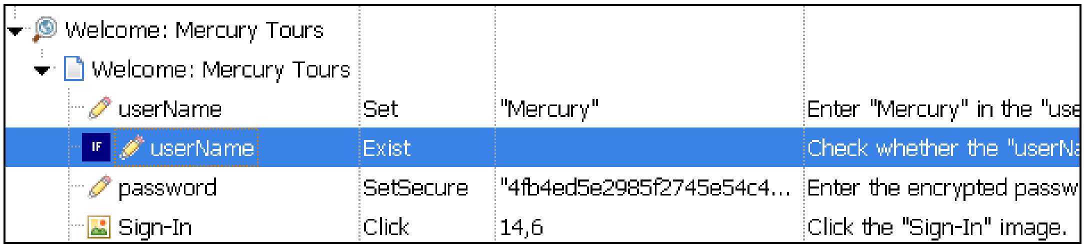
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit
IF Statement		True	Check whether (True) is true. If so:
password	SetSecure	"4fb4ed5e29...	Enter the encrypted password in the "pa
Sign-In	Click	14,6	Click the "Sign-In" image.

3. Click in the **Item** cell for the **If** statement. Then click the down arrow and select the object on which you want to perform the conditional statement. For example:



Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF userName	Set		Clear the text in the "userName" edit box. If the
password	SetSecure	"4fb4ed5e...	Enter the encrypted password in the "password"
Sign-In	Click	14,6	Click the "Sign-In" image.

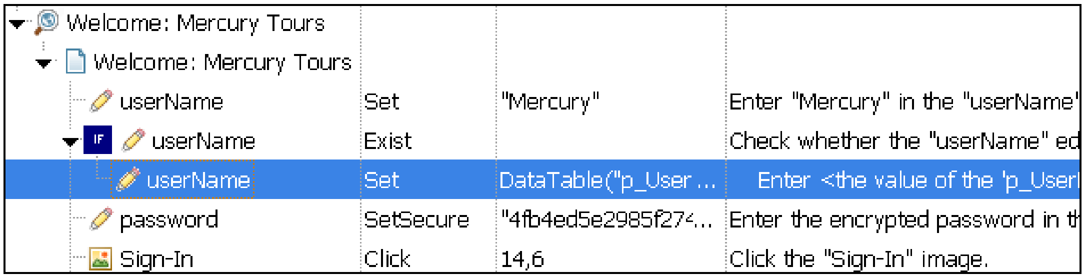
4. Click in the **Operation** cell and select the operation you want to perform. For example:



Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "use
IF userName	Exist		Check whether the "userNa
password	SetSecure	"4fb4ed5e2985f2745e54c4...	Enter the encrypted passw
Sign-In	Click	14,6	Click the "Sign-In" image.

5. If needed, click in the **Value** cell and enter the required condition. (In this example, the **Exist** property does not require a value in the **Value** cell.)

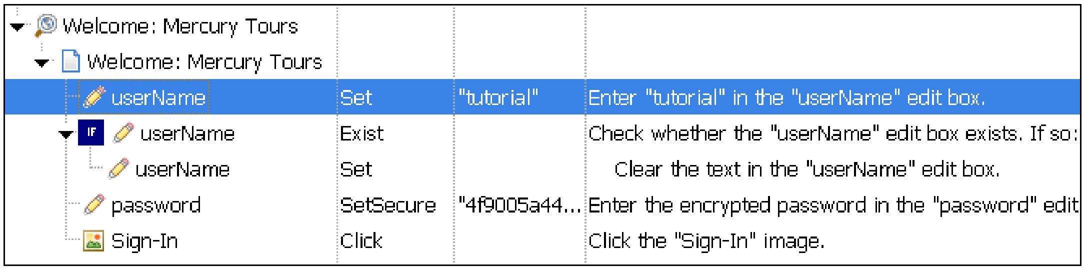
6. Insert a **Then** statement by selecting the **If** statement step and inserting a new statement (right-click and select **Insert New Step**) or by recording a new step. For example:



Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName'
IF userName	Exist		Check whether the "userName" ed
userName	Set	DataTable("p_User ...	Enter <the value of the 'p_User
password	SetSecure	"4fb4ed5e2985f27...	Enter the encrypted password in th
Sign-In	Click	14,6	Click the "Sign-In" image.

Make sure you set the values for the new step in the **Operation** and **Value** columns.

7. Delete the row immediately above the **If** statement. For example:

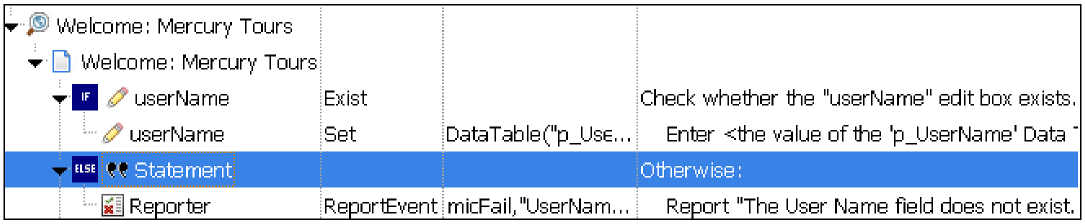


Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"tutorial"	Enter "tutorial" in the "userName" edit box.
IF userName	Exist		Check whether the "userName" edit box exists. If so:
userName	Set		Clear the text in the "userName" edit box.
password	SetSecure	"4f9005a44...	Enter the encrypted password in the "password" edit
Sign-In	Click		Click the "Sign-In" image.

8. You can now complete the statement with an **Else** statement, or you can nest an additional level in your statement. To do this, select the **If** statement and select one of the following options:

	Statement	Option
IF	**If...Then**	**Edit > Code Snippet > If...Then**
ELSE IF	**ElseIf...Then**	**Edit > Code Snippet > ElseIf...Then**
ELSE	**Else**	**Edit > Code Snippet > Else**

For example, the statements below check that the User Name edit box exists in the Mercury Tours site. **If** the edit box exists, **Then** a user name is entered; **Else** a message is sent to the Run Results.



Welcome: Mercury Tours			
Welcome: Mercury Tours			
IF userName	Exist		Check whether the "userName" edit box exists.
userName	Set	DataTable("p_Use...	Enter <the value of the 'p_UserName' Data T
ELSE Statement			Otherwise:
Reporter	ReportEvent	micFail,"UserNam...	Report "The User Name field does not exist.

The same example is displayed in the Editor as follows:

```
If Browser("Welcome: Mercury").Page("Welcome: Mercury").WebEdit("userName").Exist Then
Browser("Welcome: Mercury").Page("Welcome: Mercury").WebEdit("userName").Set DataTable ("p_UserName", dtGlobalSheet)
Else
Reporter.ReportEvent micFail, "UserName Check", "The User Name field does not exist."
End If
```

9. (Optional) After you create a conditional statement, use the **Insert Step After Block** option to insert a step outside of the conditional statement block. For details, see "Standard Steps After a Conditional or Loop Block" on page 924.

How to Insert Loop Statements In the Keyword View

Relevant for: GUI tests and scripted GUI components

To insert a loop statement in the Keyword View:

1. Select the step that you want the loop statement to follow.

2. Select **Edit > Code Snippet** and select the statement type that you want to insert from the sub-menus. The new statement is added to the Keyword View below the selected step.

3. In the **Value** column, enter the required condition, for example:
   For i = 0 to ItemsCount - 1

4. To complete the loop statement, you can:

   - Select the loop statement step and record a new step to add it to your loop statement.

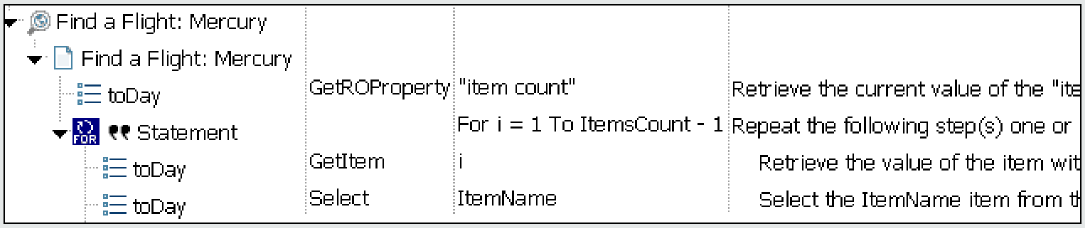
   - Select the loop statement step and right click and then select **Insert New Step** or press INSERT to insert a new step into your loop statement.

5. (Optional) After you create a loop statement, use the **Insert Step After Block** option, or press SHIFT+INSERT to insert a step outside of the loop statement block. For details, see "Standard Steps After a Conditional or Loop Block" on page 924.

Note: For details on working in the Editor, see "Programming in GUI Testing Documents in the Editor" on page 994.

Example:

The following example counts the number of items in a list and then selects them one by one. After each of the items has been selected, the test continues.



Find a Flight: Mercury			
Find a Flight: Mercury			
toDay	GetROProperty	"item count"	Retrieve the current value of the "ite
Statement		For i = 1 To ItemsCount - 1	Repeat the following step(s) one or
toDay	GetItem	i	Retrieve the value of the item wit
toDay	Select	ItemName	Select the ItemName item from t

The same example is displayed in the Editor as follows:

```
itemsCount = Browser("Welcome: Mercury").Page("Find a Flight:").
    WebList("toDay").GetROProperty ("items count")
For i = 1 To ItemsCount-1
    ItemName = Browser("Welcome: Mercury").Page("Find a Flight:").
        WebList("toDay").GetItem (i)
    Browser("Welcome: Mercury").Page("Find a Flight:").WebList("toDay").
        Select ItemName
Next
```

How to Add a Standard Step After a Conditional or Loop Block

Relevant for: GUI actions and scripted GUI components

This task includes the following steps:

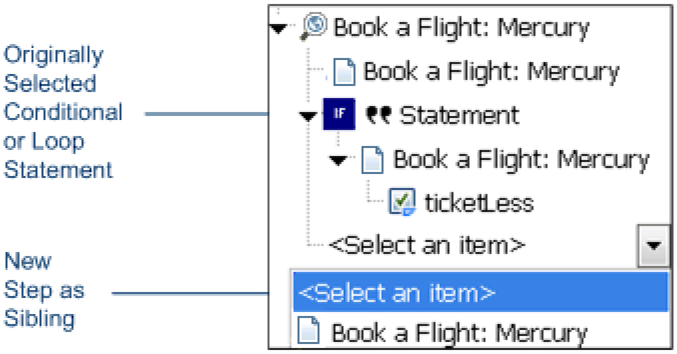
- "Select the end of the conditional or loop statement " below
- "Insert a new step using the New Step After Block option" below
- "Define the step" on the next page

1. Select the end of the conditional or loop statement

Select the conditional or loop statement step after and outside of which you want to add the new step.

2. Insert a new step using the New Step After Block option

Right-click the conditional or loop statement and select **Insert New Step After Block**, or press SHIFT+INSERT. A new step is added to the Keyword View at the end of the conditional or loop block, outside of the conditional or loop statement (as a sibling).



3. **Define the step**

   Specify the content of the step by modifying it, as described in "How to Add a Standard Step to Your Test or Component" on page 925.

How to Move an Action or Component Step

Relevant for: GUI actions and components

This task describes how to move a step to a different location within an action or component.

This task includes the following steps:

- "Drag and drop a step " below
- "Copy/cut and paste a step" below

Drag and drop a step

In the **Item** column, drag the step up or down and drop it at the required location within the action or component. When you drag a selected step, a line is displayed, enabling you to see the location to which the step will be moved.

> **For actions:** If you drag a step within its parent object, the step is displayed in the new position under its parent. If you move the step to a different parent object, the parent is duplicated, and the step is moved below it.

Copy/cut and paste a step

Copy or cut the step to the Clipboard and then paste it in the required location. (When you move, copy, or cut an action step, you also move, copy, or cut all of its sub-steps, if any.)

- **Copy.** Use **Edit > Copy** or CTRL + C.
- **Cut.** Use **Edit > Cut** or CTRL + X.
- **Paste.** Use **Edit > Paste** or CTRL + V.

Additional Guidelines for Actions:

- **Conditional and loop blocks.** (Also relevant for scripted GUI components) You cannot copy or cut only the child nodes of conditional or loop blocks. You can copy or cut them only in their entirety. After you copy or cut conditional or loop blocks to the Clipboard, you can paste them in valid locations.

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

How to Delete a Step

Relevant for: GUI actions and components

This task describes how to delete steps in the Keyword View.

This task includes the following steps:

- "Prerequisites" below

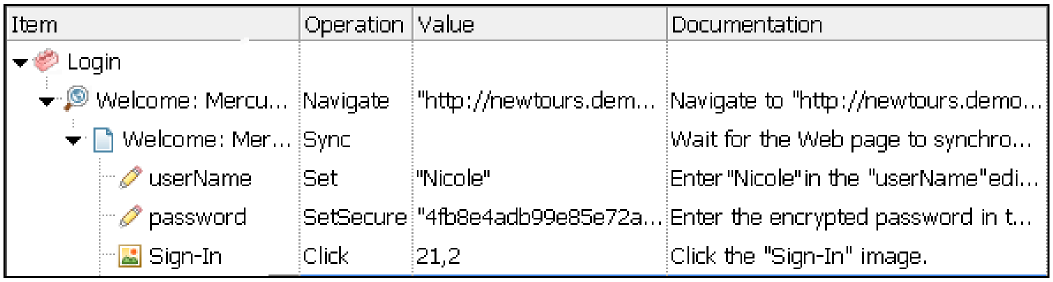
- "Delete the step " below

Note: You cannot delete a step if one of its cells is in edit mode.

1. ### Prerequisites

   - Make sure that removing the step will not prevent the action or component from running correctly.

   - **For actions:** When a step has both an operation and sub-steps defined for it, as in the example below, you can choose whether to delete only the step containing the operation of the item, or to delete the step and all of its sub-steps.



Item	Operation	Value	Documentation
Login			
Welcome: Mercu...	Navigate	"http://newtours.dem...	Navigate to "http://newtours.demo...
Welcome: Mer...	Sync		Wait for the Web page to synchro...
userName	Set	"Nicole"	Enter "Nicole" in the "userName" edi...
password	SetSecure	"4fb8e4adb99e85e72a...	Enter the encrypted password in t...
Sign-In	Click	21,2	Click the "Sign-In" image.

2. ### Delete the step

   a. Select the step that you want to delete.

b. Select **Edit > Delete**, press the DELETE key, or right-click the step and select **Delete** from the shortcut menu. A confirmation message opens.

> **For actions:** If you are deleting a reusable action that is called by another action, and if your test is stored in ALM and is using the resources and dependencies model (described on page 755), a list of the actions that call the action that you are deleting is displayed in the confirmation message box.
>
> **To copy any or all of the actions from the list to the Windows Clipboard:**
>
> **1** Select the relevant actions from the list.
>
> **2** Right-click and select **Copy Selected**, or press CTRL+C on your keyboard.

c. Click **Delete Step** to confirm. The step is deleted from the action or component.

How to Navigate in the Keyword View and Other Tips

Relevant for: GUI actions and components

This task includes the following steps:

- "General Tips" below
- "Display or Hide Columns" on the next page
- "Item Column" on the next page
- "Value Column" on the next page
- "Assignment Column (Actions Only)" on the next page
- "Comment Column (Actions Only)" on the next page
- "Comment Step (Components Only)" on page 939
- "Output Column (Components Only)" on page 939
- "Documentation Column" on page 939

General Tips

- Use the left and right arrow keys to move the focus one cell to the left or right, with the following exceptions:
  - In the **Item** column, the left and right arrow keys collapse or expand the item (if possible). If not possible, the arrow keys behave as in any other column.

  - When a cell is in edit mode (for example, when modifying a value or comment), the left and right arrow keys move the cursor within the edited cell.

- Use the standard editing commands (**Cut**, **Copy**, **Paste**, and **Delete**) in the Edit menu or in the context menu to define or modify your steps. For details, see "Keyboard Shortcuts in the Keyword View" on page 953.

- Drag and drop steps to move them to a different location within your action or component.

Display or Hide Columns

- Use the Keyword View Options dialog box. For details, see "Columns Tab (Keyword View Options Dialog Box) " on page 954.

- Right-click a column header row in the Keyword View. Then select or clear the required column name from the shortcut menu.

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

Item Column

When a row is selected (not a specific cell), you can type a letter to jump to the next row that starts with that letter.

For actions: When the entire step is selected (by clicking to its left), use the **+** key (expands a specific branch), **-** key (collapses a specific branch), and ***** key (expands all branches) to expand and collapse the **Item** tree.

Value Column

When a **Value** cell is selected, press **CTRL+F11** to open the "Value Configuration Options Dialog Box" (described on page 1579).

Assignment Column (Actions Only)

- To create or edit an assignment to or from a variable, double-click in the left part of the **Assignment** cell.

- To select either **Get from** or **Store in** (depending on whether you want to retrieve the value from a variable or store the value in a variable), click the arrow button.

- To specify or modify the name of the variable, click in the right part of the **Assignment** cell.

Comment Column (Actions Only)

- To add a comment to an existing step, select the step and type your comment in the **Comment** column.

> **Note:** You can also insert a comment step. For details, see "Insert Comment Dialog Box" on page 962.

- To modify an existing comment, double-click the comment in the **Comment** column. The cell becomes a free text field.

Comment Step (Components Only)

- To add a comment to your component, do one of the following:
  - Select **Edit > Format > Comment.**
  - Click in the **Item** cell and select **Comment** from the displayed list.
  - Right-click a component step and select **Insert Comment**. A comment row is added below the selected step.

    Enter text in the Comment row. If you do not enter text, the comment is deleted when the cursor focus is removed.
- To modify an existing comment, do one of the following:
  - Double-click the comment in the **Comment** column. The text box becomes a free text field.
  - Click the **Comment** 💬 icon, which acts as a toggle, making the comment either editable or read-only.
- To delete a comment, do one of the following:
  - Select the comment and select **Edit > Delete.**
  - Press the **DELETE** key on your keyboard.
  - Right-click and select **Delete** from the context menu.

Output Column (Components Only)

To cancel output to the parameter, click in the **Output** cell. Then click the **Cancel** button ☒ or press the **DELETE** key.

Documentation Column

- To display only the **Documentation** column, right-click the column header row and choose **Documentation Only** from the shortcut menu.
- To copy the documentation, right-click the column header row and choose **Copy Documentation to Clipboard** from the shortcut menu. Paste the documentation into a different application, as required.

How to Insert and Remove Breakpoints in the Keyword View

Relevant for: GUI actions and components

This task describes how to insert and remove breakpoints in the Keyword View. When you place a breakpoint in a step in the Keyword View, it is also displayed in the Editor (for actions and scripted components), and vice versa. For details, see "How to Use Breakpoints " on page 687.

This task includes the following steps:

- "Insert a breakpoint in the Keyword View" below
- "Remove a breakpoint from the Keyword View" on the next page

Insert a breakpoint in the Keyword View

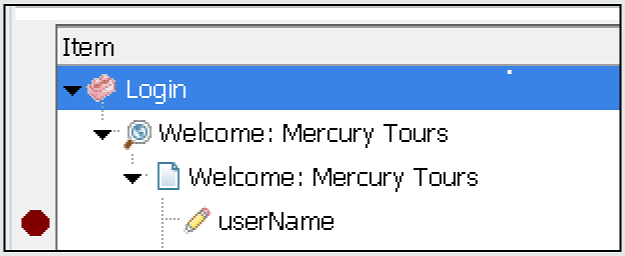
Do any of the following:

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

A red breakpoint icon ● is displayed.

Example of Inserting a Breakpoint in an Action

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



Remove a breakpoint from the Keyword View

Do any of the following:

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

How to View Properties of Step Elements in the Keyword View

Relevant for: GUI actions only

To view properties for a part of a step: Right-click the item whose properties you want to view, and select the relevant option from the context menu.

For example, you can view object properties, action properties, action call properties, checkpoint properties, and output value properties.

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

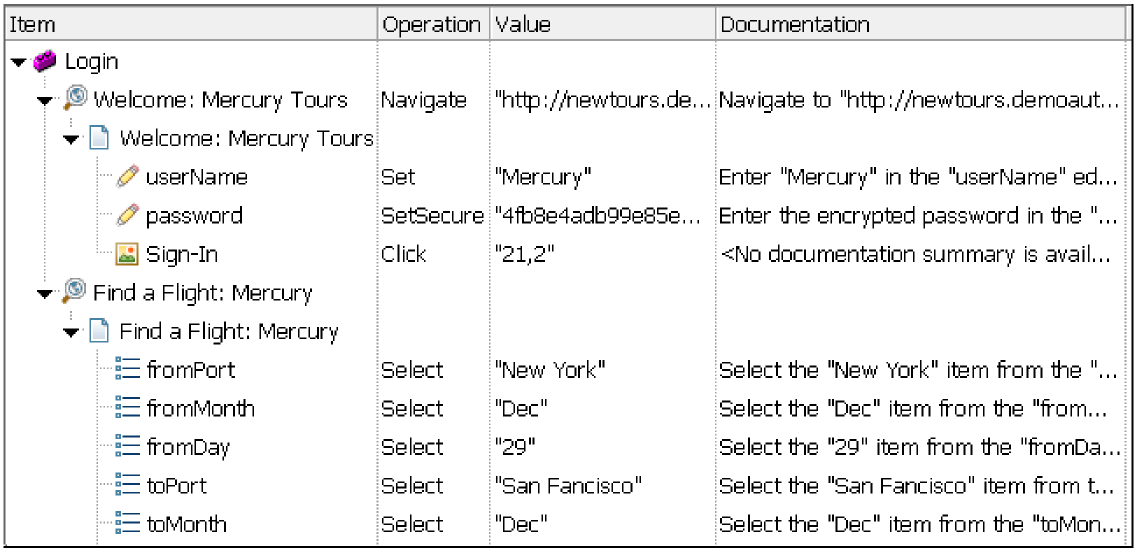
Reference

Keyword View User Interface

Relevant for: GUI actions and components

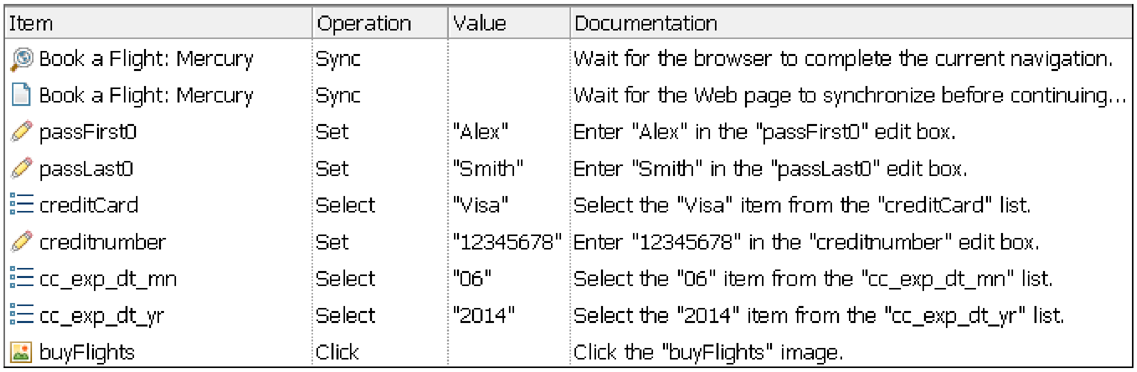
The Keyword View enables you to create and view the steps of your action or component in a keyword-driven, modular, table format. The Keyword View is comprised of a table-like view, in which each step is a separate row in the table, and each column represents the different parts of the steps. The columns displayed vary according to your selection. For details, see "Columns Tab (Keyword View Options Dialog Box) " on page 954.

The following example shows hierarchical steps in the Login action:



Item	Operation	Value	Documentation
Login			
Welcome: Mercury Tours	Navigate	"http://newtours.de...	Navigate to "http://newtours.demoaut...
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" ed...
password	SetSecure	"4fb8e4adb99e85e...	Enter the encrypted password in the "...
Sign-In	Click	"21,2"	<No documentation summary is avail...
Find a Flight: Mercury			
Find a Flight: Mercury			
fromPort	Select	"New York"	Select the "New York" item from the "...
fromMonth	Select	"Dec"	Select the "Dec" item from the "from...
fromDay	Select	"29"	Select the "29" item from the "fromDa...
toPort	Select	"San Fancisco"	Select the "San Fancisco" item from t...
toMonth	Select	"Dec"	Select the "Dec" item from the "toMon...

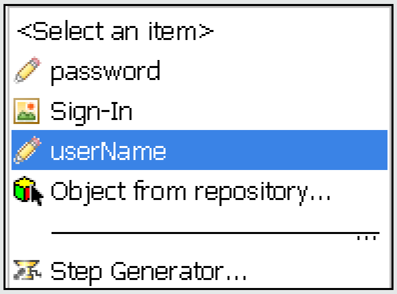
The following example shows the steps of a component:

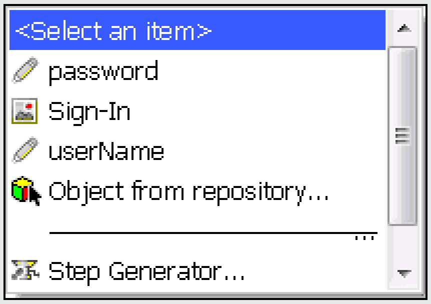


Item	Operation	Value	Documentation
Book a Flight: Mercury	Sync		Wait for the browser to complete the current navigation.
Book a Flight: Mercury	Sync		Wait for the Web page to synchronize before continuing...
passFirst0	Set	"Alex"	Enter "Alex" in the "passFirst0" edit box.
passLast0	Set	"Smith"	Enter "Smith" in the "passLast0" edit box.
creditCard	Select	"Visa"	Select the "Visa" item from the "creditCard" list.
creditnumber	Set	"12345678"	Enter "12345678" in the "creditnumber" edit box.
cc_exp_dt_mn	Select	"06"	Select the "06" item from the "cc_exp_dt_mn" list.
cc_exp_dt_yr	Select	"2014"	Select the "2014" item from the "cc_exp_dt_yr" list.
buyFlights	Click		Click the "buyFlights" image.

To access	1. Make sure that an action tab or component tab is the active document. 2. Use one of the following: ▪ **For actions or scripted components:** If the Editor is displayed, click the **Editor / Keyword View** toggle button. ▪ **For keyword components:** The Keyword View is displayed by default.
See also	• "Keyword View Context Menu Options" on page 949 • "Keyboard Shortcuts in the Keyword View" on page 953

User interface elements are described below:

UI Elements	Description
Item (actions only)	The item on which you want to perform the step. An item can be any of the following: - a **test object** from the object repository - a **utility object** - a **function call** - a **statement**, for example, a **Dim** statement - a **step generated by the Step Generator**. For details, see "How to Insert Steps Using the Step Generator" on page 978. This column displays a hierarchical icon-based tree. The highest level of the tree are actions, and all steps are contained within the relevant branch of the tree. Steps performed within the same parent object are displayed under that same object. Function calls, utility objects, and statements are placed in the tree hierarchy at the same level as the item above them (as a sibling). **Example of Item list** The test objects available in the **Item** list are the sibling and child test objects of the previous step's test object, as defined in the object repository. The example below shows the objects available for the step following a **userName** test object.  <Select an item> password Sign-In userName Object from repository... ... Step Generator... For task details, see "How to Select an Item for Your Step " on page 926.

UI Elements	Description
Item (components only)	The item on which you want to perform the step. An item can be any of the following: • a **test object** from the object repository • a user-defined function (**Operation**), for example, a function that opens an application at the start of a business component or checks the value of a specific property. Available only if user-defined functions were added to a function library that is associated with the component's application area. For details, see "Associated Function Libraries" on page 1031 and "User-Defined Functions and Function Libraries" on page 1029. • **Comment.** Free text cell that spans the entire row. After a step is converted into a Comment step it cannot be restored to another type of step. You use the Comment option to enter manual steps or to provide information on adjacent steps. For details, see "Comments in the Keyword View" on page 922. You must select an option from the **Item** column before you can add additional content to a step. **Example of Item list** The test objects available in the **Item** list are the sibling and child test objects of the previous step's test object, as defined in the object repository. The example below shows the objects available for the step following a **userName** test object.  <Select an item> password Sign-In userName Object from repository... Step Generator... For task details, see "How to Select an Item for Your Step " on page 926.

UI Elements	Description
Operation	The operation to be performed on the item. This column contains a list of all available operations (methods, functions, or properties (or sub-procedures for components)) that can be performed on the item selected in the **Item** column, for example, **Click** and **Select**. For task details, see "How to Select an Operation for Your Step " on page 927. **For actions:** The **Operation** column displays the default operation. **For components:** The **Operation** column displays the most commonly used operation for the item. If the application area used by a component is associated with at least one function library, the functions from that function library are accessible from this column. You specify function libraries in the Function Libraries pane of the Application Area. For details, see "Function Libraries Pane (Application Area)" on page 2102.

UI Elements	Description
Value	The argument values for the selected operation. For actions, this can display the content of the statement. The **Value** cell is partitioned according to the number of arguments of the selected option in the **Operation** column. If an argument has a predefined list of values, a drop-down list of possible values is provided, and you cannot manually enter a value in this box. For task details, see "How to Define Values for Your Step Arguments " on page 928. • If you click in the Value cell for an operation that accepts parameters, the parameterization icon <#> is displayed. Click this icon to open the "Value Configuration Options Dialog Box". • If you click in the Value cell of a checkpoint step, the checkpoint properties icon ☑ is displayed. Click this icon to open the relevant Checkpoint Properties dialog box. **For components:** An argument value can be a constant, a **local** parameter, or a **component** parameter, depending on the selected option. • **Local parameter.** A local parameter is specific to the business component and can only be accessed by that component. It is intended for use in a single step or between component steps, for example, as an output parameter for one step and an input parameter for a later step. For details, see "How to Define Values for Your Step Arguments " on page 928. • **Component parameter.** A component parameter is a parameter that can be accessed by any component in your ALM project. For details, see "Parameters Tab (Properties Pane - GUI Testing)" on page 394. If you click in the **Value** cell after you specify a local or component parameter for it, an icon specifying the type of parameter is displayed in the cell: 🧩 Indicates a component parameter. 🏠 Indicates a local parameter.

UI Elements	Description
Assignment (actions only)	The assignment of a value to or from a variable. For example, **Store in cCols** would store the return value of the current step in a variable called cCols, which you could then use later in the test. You can select either **Store in** or **Get from**, depending on whether you want to retrieve the value from a variable or store the value in a variable. • **Store in X.** Value is equivalent to an **X = <*step*>** line in the Editor. • **Get From X.** Value is equivalent to a **<*step*> = X** line in the Editor. For details on storing variables, see "Storage Location Options Dialog Box" on page 991.
Output (components only)	The parameter in which output values for the step are stored. For details, see "Output Value for Your Component Step - Output Column" on page 923. **Example:** If you select an output parameter named cCols, the output value of the current step would be stored in the **cCols** parameter. You can then use the value stored in the output parameter later in the component as an input parameter. As in the **Value** column, you can use two types of parameters when specifying an output parameter—a local parameter or a component parameter. If you click in the **Output** cell after you specify an output parameter for it, an icon specifying the type of parameter is displayed in the cell: 🧩 Indicates a component parameter. 🏠 Indicates a local parameter.
Comment	**For actions:** A free text edit box for any information you want to add regarding the step. These are also displayed as inline comments in the Editor. **For components:** A manual step or other free textual notes between steps. A comment is displayed as an entire row (not in a column). For details, see "Comments in the Keyword View" on page 922. **Note:** • UFT does not process comments during a run session. • After you insert a comment, you cannot change it to a step.

UI Elements	Description
Documentation	Read-only auto-documentation of what the step does in an easy-to-understand sentence, for example, Click the "Sign-in" image. or Select "San Francisco" in the "toPort" list. **Tip:** • Displaying only this column (and hiding the other columns) is useful if you want to print or view manual testing instructions. For details, see "How to Navigate in the Keyword View and Other Tips" on page 937. • If you created a function library and associated it with a test or with a component's associated application area, documentation for its functions can be displayed only if you defined the relevant text in the function library. For details, see "Add documentation details to the function - optional" on page 1053 and "User-Defined Functions and Function Libraries" on page 1029.

Keyword View Context Menu Options

Relevant for: GUI tests and components

To access	1. Do one of the following: ▪ Ensure that a GUI test 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 **View > Keyword View** or click the **Keyword View** toggle button.
Relevant tasks	"How to Add a Standard Step to Your Test or Component" on page 925
See also	• "Keyword View Overview" on page 921 • "Keyword View User Interface" on page 942

Context menu options for GUI actions and scripted GUI components

Menu Command	Description
Action Properties (action level only)	Opens the "Action Properties Dialog Box" (described on page 900), enabling you to specify options, parameters, and associated object repositories for a stored action.
Action Call Properties (action level only)	Opens the "Action Call Properties Dialog Box" (described on page 895), enabling you to specify the number of run iterations according to the number of rows in the data table, and to define the values of input parameters and the storage location of output parameters.
Object Repository (action level only)	Opens the "Object Repository Window" (described on page 1274), which displays a tree containing all objects in the current action.
Object Properties	Opens the "Object Properties Dialog Box" (described on page 1271), which displays the read-only properties of the selected object. **Note:** This command is only available when the selected step contains a test object.
Comment Properties	Opens the "Comment Properties Dialog Box" (described on page 963), which enables you to view the properties of the selected comment. **Note:** This command is only available when the selected step is a comment step.
Checkpoint Properties	Opens the relevant Checkpoint Properties dialog box for a selected object. **Note:** This command is only available when the selected step is a checkpoint step.
Output Value Properties	Opens the relevant Output Value Properties dialog box for a selected object. **Note:** This command is only available when the selected step is an output value step.
Insert Standard Checkpoint	Opens the "Checkpoint Properties Dialog Box" on page 1432 (described on page 1432), enabling you to create a checkpoint on the selected test object.

Menu Command	Description
Insert Output Value	Opens the "Output Value Properties Dialog Box" on page 1498 (described on page 1498), enabling you insert an output value step on a selected test object.
Insert New Step	Inserts a new step into your action.
Insert New Step After Block	Inserts a new step after a conditional or loop block into your action.
Insert Step > Step Generator	Opens the "Step Generator Dialog Box" (described on page 983).
Insert Step > Comment	Inserts a Comment step into your action.
Insert Step > Report	Opens the Report details dialog, enabling you to add details to be included in the report of the test.
Conditional Statement > If...Then	Inserts an **If...Then** conditional statement step.
Conditional Statement > ElseIf...Then	Inserts an **ElseIf...Then** conditional statement step
Conditional Statement > Else	Inserts an **Else** conditional statement step.
Loop Statement > While...Wend	Inserts a **While...Wend** loop statement step.
Loop Statement > For...Next	Inserts a **For...Next** loop statement step.
Loop Statement > Do...While	Inserts a **Do...While** loop statement step.
Loop Statement > Do...Until	Inserts a **Do...Until** loop statement step.

Menu Command	Description
Start Transaction	Inserts a **StartTransaction** step into your action, marking the beginning of the transaction to be timed. (Relevant only if the test includes transactions to be used by LoadRunner or Business Availability Center.)
End Transaction	Inserts an **EndTransaction** step in the test, marking the end of the transaction to be timed. (Relevant only if the test includes transactions to be used by LoadRunner or Business Availability Center.)
Action > Insert Call to New	Opens the "Insert Call to New Action Dialog Box (GUI Testing)" (described on page 913), enabling you to create a new action.
Action > Insert Call to Copy	Opens the "Select Action Dialog Box " (described on page 915), which inserts a call to an editable copy of an existing action.
Action > Insert Call to Existing	Opens the "Select Action Dialog Box " (described on page 915), which inserts a call to an existing reusable action.
Insert/Remove Breakpoint	Sets or clears a breakpoint in your action or component.
Enable/Disable Breakpoint	Enables or disables the selected breakpoint in your test.
Run from Step	Starts a run session from the selected step.
Debug from Step	Runs from the selected step instead of the start of the test or component.
Run to Step	Runs a test or component until the current step.
Optional Step	Inserts an optional step into your action or component.
Cut	Removes the currently selected line or text from your action or component.
Copy	Copies the currently selected step or text.
Paste	Pastes the currently copied text.
Delete	Deletes the current step or text.
Expand Sub Tree	Expands all substeps in the current action.
Collapse Sub Tree	Collapses all substeps in the current action.

Context menu options for keyword GUI components

Menu Command	Description
Insert New Step	Inserts a new step into your component.
Insert Comment	Inserts a Comment step into your component.
Insert Operation	Inserts a new operation (function) step into your component.
Cut	Removes the currently selected line or text from your component.
Copy	Copies the currently selected step or text.
Paste	Pastes the currently copied text.
Delete	Deletes the currently step or text.

Keyboard Shortcuts in the Keyword View

Relevant for: GUI actions and components

If you prefer to use your keyboard to edit an action or a component, you can use the following keyboard shortcuts to navigate within the Keyword View:

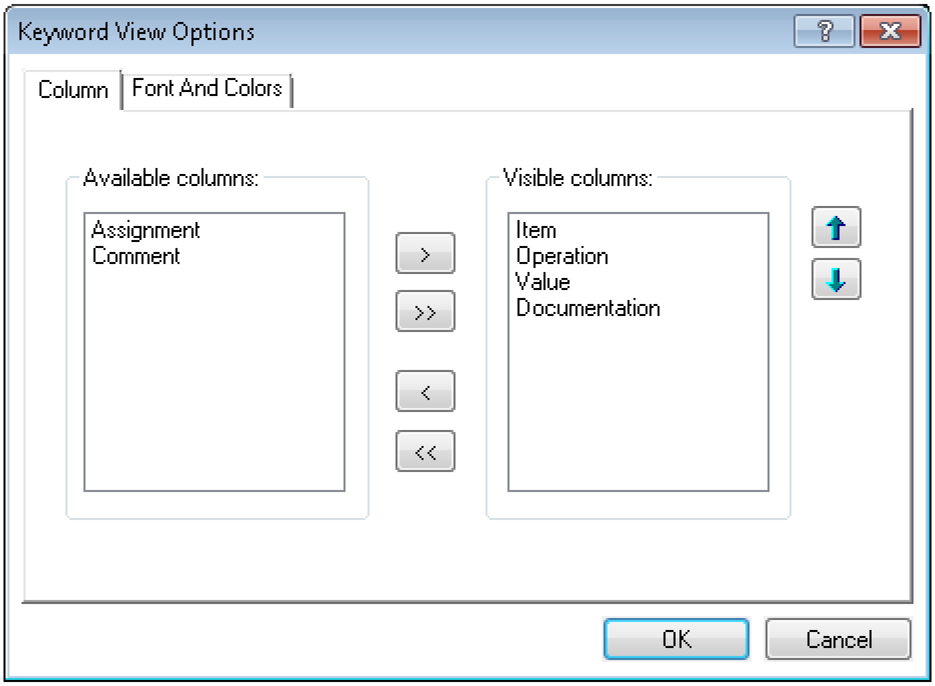
- **Actions and scripted components:** Press SHIFT+INSERT to add a new step after a conditional or loop block.
- Press INSERT to add a new step below the currently selected step.
- Use the TAB and SHIFT+TAB keys to move the focus left or right within a single row, unless you are in a cell that is in edit mode. If so, press ENTER to exit edit mode, and then you can use the TAB keys.
- When the value cell of a checkpoint step is selected, press CTRL+F11 to open the relevant Checkpoint Properties dialog box.
- When a cell containing a list is selected, you can:
  - Press SHIFT+F4 to open the list for that cell.
  - Change the selected item by using the up and down arrow keys. In the **Item** column, the list must be open before you can use the arrow keys.
  - Type a letter or sequence of letters to move to a value that starts with the typed letters. The typed sequence is highlighted in white.
- **Actions and scripted components:** Press F7 to use the Step Generator to add a new step below the selected step.

Columns Tab (Keyword View Options Dialog Box)

Relevant for: GUI actions and components

This tab enables you to specify the columns you want to display in the Keyword View. You can also specify the order in which the columns are displayed.

The following image shows the options available for an action. Options for a component are similar.



To access	In the Keyword View, right-click a column header and select **View Options.**
Important Information	• You can use standard SHIFT and CTRL key behaviors to select multiple column names in the **Available columns** or **Visible columns** list. • As an alternative to using this dialog box tab, you can: ▪ Select which Keyword View columns to display by right-clicking the column header row in the Keyword View and selecting the columns to display or hide. ▪ Select the **Documentation Only** item in the displayed menu. This is useful if you want to print the documentation column information for use as instructions for manual testing.
See also	• "Keyword View User Interface" on page 942 • "Fonts and Colors Tab (Keyword View Options Dialog Box)" on the next page

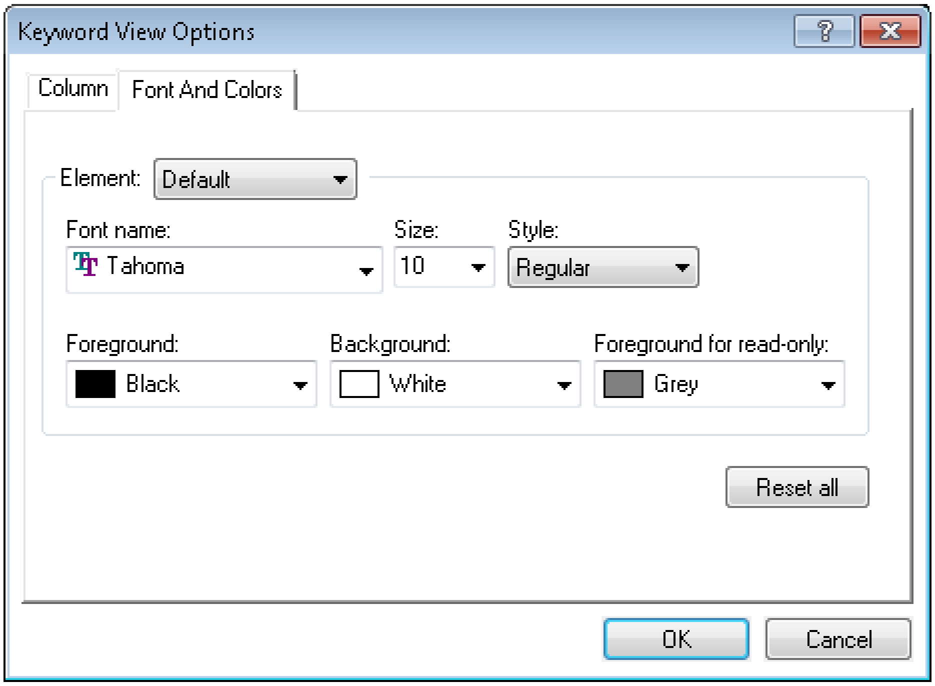
User interface elements are described below:

UI Elements	Description
Available columns	Columns not currently displayed in the Keyword View.
Visible columns	Columns currently displayed in the Keyword View.
>	**Display Selected Columns.** Moves the selected columns from the **Available columns** list to the **Visible columns** list. Alternatively, you can double-click an item to move it to the other list.
>>	**Display All Columns.** Moves all columns from the **Available columns** list to the **Visible columns** list.
>	**Hide Selected Columns.** Moves the selected columns from the **Visible columns** list to the **Available columns** list. Alternatively, you can double-click an item to move it to the other list.
>>	**Hide All Columns.** Moves all columns from the **Visible columns** list to the **Available columns** list.
↑ ↓	**Move Up / Move Down.** Adjusts the order in which the selected columns appear in the Keyword View. **Note:** The order of the columns in the Keyword View does not affect the order in which the cells need to be completed for each step. For example, if you choose to display the **Operation** column to the left of the **Item** column, you must still select the item first. Only then is the **Operation** column list refreshed to match the selection you made in the **Item** column.

Fonts and Colors Tab (Keyword View Options Dialog Box)

Relevant for: GUI actions and components

This tab enables you to specify text and color display options for different elements in the Keyword View.



Element: Default

Font name: Tahoma

Size: 10

Style: Regular

Foreground: ■ Black

Background: □ White

Foreground for read-only: ■ Grey

To access	1. In the Keyword View, right-click a column header and select **View Options.** 2. In the Keyword View Options dialog box, select the **Fonts and Colors** tab.
See also	- "Keyword View User Interface" on page 942 - "Columns Tab (Keyword View Options Dialog Box) " on page 954

User interface elements are described below:

UI Elements	Description
Element	You can specify different font and color options for each of these Keyword View elements. Select one of the following elements to see the current definitions and modify them: • **Alternate Rows.** The background color of every other row. The font and text color for the alternate rows is the same as the font and text color defined for the **Default** element. • **Comment.** The row and text of comment lines. Note that all of the available formatting options apply to entire comment rows, not to comments within a regular step row. For comments within a step row, only the specified **Foreground** color applies (all other settings are taken from the **Alternate Rows**, **Default**, or **Selected Row** settings, as appropriate). • **Default.** All rows and text in the Keyword View (except for the elements listed below). • **Selected Row.** The row and text currently selected (highlighted).
Font Name	Enables you to modify the font used for text in the selected element. You cannot change the font for **Alternate Rows** or **Selected Row** elements. **Note:** When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your action or component may not be correctly displayed in the Keyword View. However, the action or component will still run in the same way, regardless of the font you choose.
Size	Enables you to modify the font size used for text in the selected element. You cannot change the font size for **Alternate Rows** or **Selected Row** elements.
Style	Enables you to modify the font style used for text in the selected element. You can select **Regular**, **Bold**, **Italic**, or **Underline** font styles. You cannot change the font style for **Alternate Rows** or **Selected Row** elements.
Foreground	Enables you to modify the text color for the selected element. You cannot change the foreground color for **Alternate Rows**.
Background	Enables you to modify the row color for the selected element.
Foreground for read-only	Enables you to modify the text color for rows that are read-only. This option cannot be changed for **Alternate Rows**.
Reset all	Resets all Fonts and Colors tab options to the default settings.

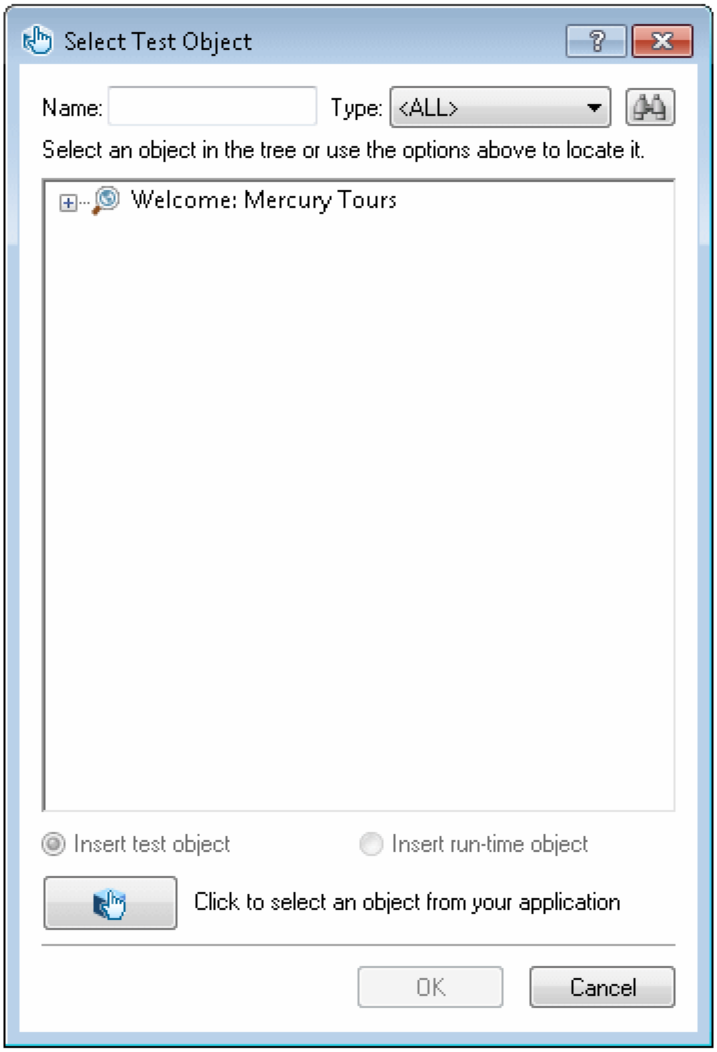
Select Test Object Dialog Box

Relevant for: GUI actions and components

This dialog box displays the object repository tree and enables you to:

- Select an object in the object repository tree for your step.
- Select an object from your application for your step. This adds a test object to the local object repository.
- Select a related object to use in a visual relation identifier. This adds a test object to the same object repository as the test object you want to identify. For details on visual relation identifiers, see "Visual Relation Identifiers" on page 1209.
- **For actions:** Enter a .Object statement for the selected test object in your test.

The following image shows an example of this dialog box when opened from the Keyword View for an action or from the Visual Relation Identifier dialog box. Similar options are available for components or when opened from the Step Generator.


Select Test Object
Name:
Type: <ALL>
Select an object in the tree or use the options above to locate it.
Welcome: Mercury Tours
○ Insert test object
○ Insert run-time object
Click to select an object from your application
OK
Cancel

To access	Use one of the following: • In the **Item** cell of the Keyword View, click the arrow button to display the **Item** list and then select **Object from repository.** • In the "Visual Relation Identifier Dialog Box" (described on page 1251), click the **Add** button. • **For actions and scripted components only:** In the "Step Generator Dialog Box" (described on page 983), select **Test Objects** from the **Category** box list and click the **Select Object** button .

Important information	• You can select any object in the object repository tree for your new step. For more information on the object repository, see "Managing Test Objects in Object Repositories" on page 1198. • If the object that you want to use in the new step is not in the object repository, you can select an object in your application by using the pointing hand. This adds it to the local object repository. • If you select an object in your application that is not in an associated shared object repository, a test object is added to the local object repository when you insert the new step. After you add a new test object to the local object repository, it is recommended to rename it, if its name does not clearly indicate its use. For example, you may want to rename a test object named Edit (that is used for entering a username) to UserName. This will enable other users to select the appropriate test object when adding steps using test objects located in this shared object repository. • After you add the required objects to the local object repository, you can use the Object Repository Merge Tool to update the associated shared object repository and make the new test objects available to other actions and components. For details, see "When to Merge Shared Object Repositories" on page 1352. • If you are adding a container test object, it is also recommended to specify its context, for example, if you are adding a confirmation message box from a Login page, you may want to name it Login > Confirm. For details, see "Renaming Test Objects " on page 1206. • **For components:** ALM users can select only objects that are stored in the shared object repositories (in the component's application area).
Relevant tasks	• "How to Insert Steps Using the Step Generator" on page 978 • "How to Select an Item for Your Step " on page 926
See also	• "Managing Test Objects in Object Repositories" on page 1198 • "Native Properties and Operations" on page 1008 • "Visual Relation Identifier Dialog Box" on page 1251

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

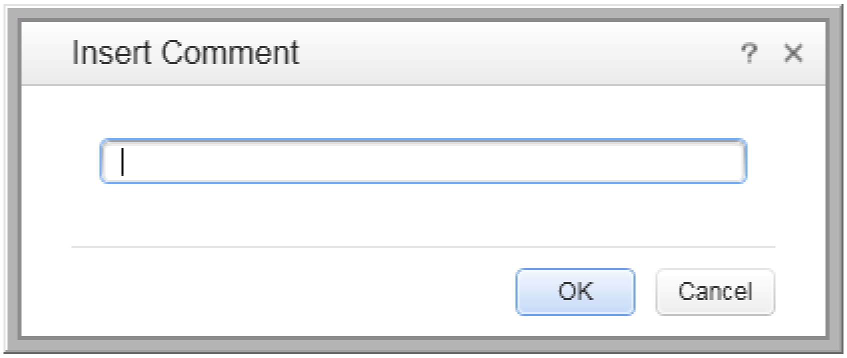
UI Elements	Description
🔍	**Find Next.** Searches from the currently selected node, displaying all objects that match your criteria. The first object in the list that matches your criteria is highlighted. You can click the **Find Next** button to navigate through all the objects that match your search criteria. The search continues to the end of the tree, then wraps to the beginning of the tree, and continues. **Tip:** Press **F3** to find the next object that matches your search criteria, or **SHIFT+F3** to find the previous match.
👆	**<Pointing hand>.** Enables you to select an object from your application and add it to the shared object repository so that you can use it in your action or component steps. This is useful if the shared object repository does not include the test object that you need for a particular step. Click on the required object in your application to add it to the shared object repository tree. In some environments, as you move the pointer over your application, the available test objects are highlighted. For details, see "Tips for Using the Pointing Hand" on page 1196.
Name	The full or partial name of the object to find. This option is useful if the object repository is very large. If you leave the box empty, all objects that match the selected object **Type** are displayed. **Example:** Enter **p** to search for all object names containing the letter p.
Type	The type of object for which to search. Specify a type or select **<All>** to search for the object in all the object types. The object types in this list are a generic grouping of objects according to the general object characteristics. For example, the **List** type contains list and list view objects, as well as combo boxes; the **Table** type contains both tables and grids; and the **Miscellaneous** type contains a variety of other objects, such as WebElement and WinObject. **Example:** Suppose you want to add a password object that you know is an Edit box. You can search all the Edit type objects for one called password, or even one containing the letter p.
<object repository tree>	List of objects in the object repositories associated with the current action or component. You can select a test object from this tree to insert into a step.

UI Elements	Description
Insert test object	Inserts the selected test object in the step. **Note:** This option is available only when this dialog box is opened from the Keyword View for an action or from the Visual Relation Identifier dialog box.
Insert run-time object	Inserts an Object statement step for the selected test object. (Relevant for any object that supports the .Object property.) **Note:** This option is available only when this dialog box is opened from the Keyword View for an action or from the Visual Relation Identifier dialog box.

Insert Comment Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to insert a new comment for a step.



To access	In the Keyword View, do one of the following: • Select **Design > Add Comment**. • Right-click any step that does not yet contain a comment and select **Insert Step > Comment** (actions) or **Insert Comment** (keyword components).

Important information for tests and scripted components	• If the **Comment** column is not visible, right-click any column header and select **Comment.** • You can add comments directly in the Keyword View by selecting a step and typing the comment in the **Comment** column. • You can modify comments at any time directly in the Keyword View, or by using the Comment Properties dialog box. • To add a comment in the Editor, type an apostrophe (') and then type your comment. You can add a comment at the end of a line or at the beginning of a separate line. • If you want to add the same comment to every action that you create, you can add the comment to an action template. For details, see "How to Manage the Structure of Your GUI Test" on page 888.

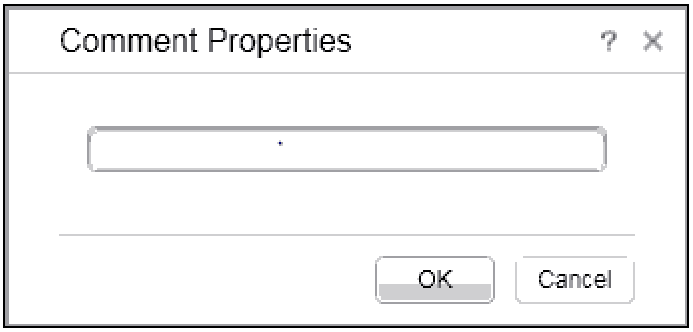
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<comment text area>	The comment text to enter.

Comment Properties Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to modify the text of a selected comment.



To access	In the Keyword View, right-click any step containing a comment and select **Comment Properties**.
Important information	• If the **Comment** column is not visible, right-click any column header and select **Comment.** • In the Keyword View, you also can modify the comment text directly in the **Comment** column. • In the Editor, you can manually edit the text to overwrite any existing comment.

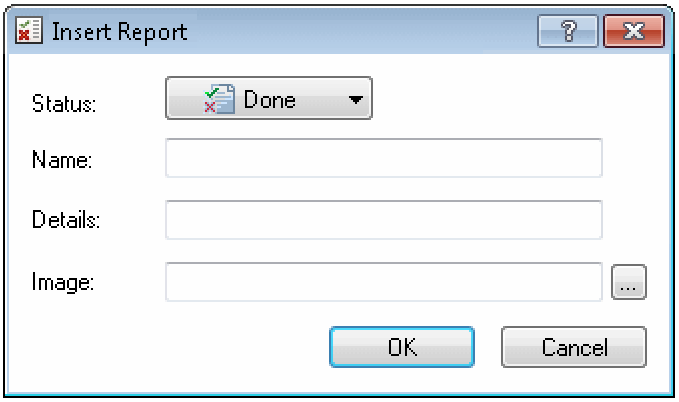
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<comment text area>	The comment text to modify.

Insert Report Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to define a message that UFT sends to your run results.



To access	In the Keyword View: • Select a step and select **Edit > Report Step**. • Right-click a step and select **Insert Step > Report**.
Important Information	When you add a report step, a **Reporter.ReportEvent** step is inserted.
See also	"Message Statements" on page 975

User interface elements are described below:

UI Elements	Description
Status	The status that will result from this step from the **Status** list: • **Passed.** Causes this step to pass. Sends the specified message to the report. • **Failed.** Causes this step (and therefore the test itself) to fail. Sends the specified message to the report. • **Done.** Sends a message to the report without affecting the pass/fail status of the step. • **Warning.** Sends a warning status for the step, but does not cause the test to stop running, and does not affect its pass/fail status.
Name	The name for the step, for example, Password edit box. This name is displayed in the Run Results tree as the node label for this step.
Details	The description of this step. For example, Password edit box does not exist. The description is displayed in the upper-right pane of the Run Results Viewer.
Image	The name of an image to include in the run results with this step. The image is displayed in the bottom-right pane of the Run Results Viewer. The image can be any of the supported file types, such as BMP, JPEG, GIF, TIF, or PNG. **Note:** • You cannot specify an image stored in ALM. • Including large images in the run results may impact performance. • If an image is specified as a relative path, UFT will first search the Results folder for the image and then the search paths specified in the **Folders** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node).

Password Encoder Tool

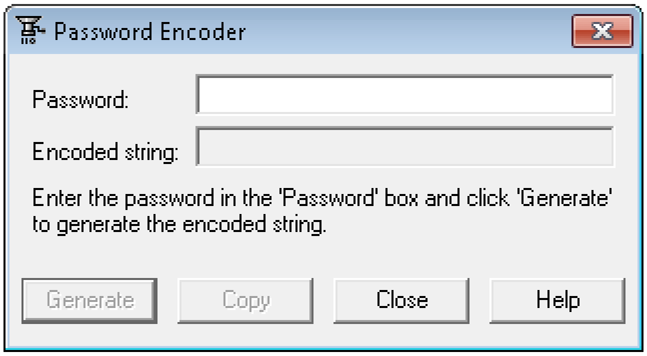
Relevant for: GUI actions and components

This tool enables you to encode passwords so that you can use the resulting strings in the following ways:

For tests: method arguments or data table parameter values (enabling you to place secure values into the data table)

For components: operation arguments

For example, your Web site may include a form in which the user must supply a password. You may want to test how your site responds to different passwords, but you also want to ensure the integrity of the passwords.



Enter the password in the 'Password' box and click 'Generate' to generate the encoded string.

To access	From the **Windows** menu, select **Start > Programs > HP Software > HP Unified Functional Testing > Tools > Password Encoder**. **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.
Important information for Business Process Testing	Use only if you are working with Quality Center. If you are working with ALM, you must encode passwords in ALM. Otherwise, during a run session, encoded strings are treated as plain text. For details on encoding, see the *HP Business Process Testing User Guide*.
See also	**For tests:** You can also encrypt strings in Data pane cells using the Encrypt option in the Data menu. For details, see "Data Menu (GUI Testing Data Pane)" on page 243.

User interface elements are described below:

UI Elements	Description
Password	The password to encrypt.
Encoded string	The encrypted string that is generated when you click **Generate**.
Generate	Encrypts the string you entered in the **Password** box.
Copy	Copies the encrypted password to the Clipboard. You can then paste it as value for a method or operation argument, or you can insert it as a secure value in the Data pane (tests only).

Troubleshooting and Limitations - Keyword View

Relevant for: GUI actions and components

This section describes troubleshooting and limitations for the Keyword View.

If you use the **Object** property in a step in the Keyword View, it may take a long time for UFT to retrieve the object information from the application. This may affect UFT's response time when you open and select from the various drop-down lists in the step.

Workaround: Use the Editor when working with the **Object** property.

Chapter 35: Generated Programming Operations

Relevant for: GUI tests and scripted GUI components

This chapter includes:

Concepts

Programming Statements Overview

Relevant for: GUI tests and scripted GUI components

When you design tests, you usually begin by adding steps that represent the operations an end-user would perform as part of the business process you want to test. Then, to increase the power and flexibility of your test, you can add steps (programming statements) that contain programming logic to the basic framework.

Programming statements can contain:

- **Test object operations.** These are methods and properties defined by UFT. They can be operations that a user can perform on an object, operations that can retrieve or set information, or operations that perform operations triggered by an event.
- **Native operations.** These are methods and properties defined within the object you are testing, and therefore are retrieved from the run-time object in the application.
- **VBScript programming commands**. These affect the way the test runs, such as Conditional Statements and Synchronization Points. These are often used to control the logical flow of a test.
- **Comments**. Use comments to explain sections of your tests to improve readability and to make them easier to update. A comment is an explanatory remark in a program, and does not get processed when UFT runs. For details, see "Insert Comment Dialog Box" on page 962.

This section also includes:

Step Generator

Relevant for: GUI tests and scripted GUI components

The Step Generator enables you to add steps by selecting from a range of context-sensitive options and entering the required values, so that you do not need to memorize syntax or to be proficient in high-level VBScript. You can use the Step Generator from the Keyword View and also from the Editor.

In the "Step Generator Dialog Box" (described on page 983) you can define steps that use:

- Test object operations (tests only).
- Utility object operations.
- Calls to library functions (tests only), VBScript functions, and internal script functions.

For example, you can add a step that checks that an object exists, or that stores the returned value of a method as an output value or as part of a conditional statement. You can parameterize any of the values in your step.

When you insert a new step using the Step Generator, it is added to your test after the selected step, and the new step is selected. For details on the hierarchy of steps and objects and the positioning of new steps, see "Keyword View User Interface" on page 942.

For more details, see "Step Generator Dialog Box" on page 983.

Conditional Statements

Relevant for: GUI tests and scripted GUI components

You can control the flow of your test with conditional statements. Using conditional **If...Then...Else** statements, you can incorporate decision-making into your tests.

The **If...Then...Else** statement is used to evaluate whether a condition is true or false and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. The following comparison operators are available: less than **<**, less than or equal to **<=**, greater than **>**, greater than or equal to **>=**, not equal **<>**, and equal **=**.

Your **If...Then...Else** statement can be nested to as many levels as you need.

The statement has the following syntax:

If *condition* **Then** *statements* [**Else***elsestatements*] **End If**

Or, you can use the block form syntax:

```
If condition Then
    [statements]
[ElseIf condition-n Then
    [elseifstatements] . . .
[Else
    [elsestatements]
End If
```

For more details:

- On inserting conditional statements using the Step Generator, see "Step Generator Dialog Box" on page 983.
- On working with conditional steps in the Editor, see "Comments, Control-Flow, and Other VBScript Statements " on page 1006, and the VBScript documentation (select **Help > HP Unified Functional Testing Help > VBScript Reference**).
- On working with conditional steps in the Keyword View, see "Conditional and Loop Statements in the Keyword View" on page 923.

Example:

The following example checks whether a valid order number is entered in the Order No. box.

To do this, UFT activates the Open Order dialog box (brings it into focus), selects the Order No. check box, and opens a box in which the user inserts a value—the relevant order number—and clicks **OK**. The first conditional statement instructs UFT to verify that the value entered by the user is greater than zero. **If** it is not greater than zero, a message box is displayed, stating that the value entered is invalid. When the user clicks **OK** to close the message box, the run session ends.

Otherwise, if the value entered is greater than zero, UFT inserts the above value in the Order No. box.

The next **If** statement instructs UFT to check whether the order number exists in the application and to send a report to the Run Results indicating that the step passed or failed. If the step failed because the order number was invalid, the Flight Reservations error message opens, and UFT clicks **OK** to close this message box before ending the run session.

Note: Many of the lines in the example below are comments.

For example:

```
`Set the focus on (activate) the Open Order dialog box
Window("Flight Reservation").Dialog("Open Order").Activate
`Insert a check mark in the Order No. check box
Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").Set "ON"

Insert an order number in the displayed box and save the value as "OrderNo" for use later in the script.
If the value is less than or equal to 0, generate a messagebox.
(If the value is illegal and a message box is generated, end the run session when the user clicks OK.)
OrderNo = InputBox("Enter Order Number")
If OrderNo <= 0 Then
 Msgbox "You entered an invalid order number."
 ExitAction
End If
`Insert the saved order number value in the Order No. box
Window("Flight Reservation").Dialog("Open Order").WinEdit("OrderNumberEdit").Set OrderNo
`Click OK to close the Open Order dialog box
```

```
Window("Flight Reservation").Dialog("Open Order").WinButton("OK").Click
`Check if an error message opens and send a report to the run results
If Window("Flight Reservation").Dialog("Open Order").Dialog("FlightReservations").Exist Then
  Reporter.ReportEvent micFail, "Check that the value of the order number is legal", "The order n
umber does not exist."
  Window("Flight Reservation").Dialog("Open Order").Dialog("FlightReservations").WinButton("O
K").Click
Else
  Reporter.ReportEvent micPass, "Check that the value of the order number is legal", "The order n
umber exists."
End If
```

With Statements

Relevant for: GUI tests and scripted GUI components

With statements make your script (in the Editor) more concise and easier to read by grouping consecutive statements with the same parent hierarchy.

In addition, using **With** statements might help your script run faster. When running a **With** statement, UFT identifies the object in the application before running the first statement, but does not re-identify it before running each statement.

On the other hand, this can affect the running of your test if the object referenced by the **With** statement is refreshed, redrawn, or changed in some way in the application while running the **With** statement. To instruct UFT to re-identify the object in the application before running the next statement, add a statement that calls the **RefreshObject** test object operation. For details on the **RefreshObject** operation, see the *HP UFT Object Model Reference for GUI Testing*.

The **With** statement has the following syntax.

```
With object
      statements
End With
```

Item	Description
object	An object or a function that returns an object.
statements	One or more statements to be performed on an object.

Example

For example, you could replace this script:

```
Window("Flight Reservation").WinComboBox("Fly From:").Select "London"
Window("Flight Reservation").WinComboBox("Fly To:").Select "Los Angeles"
Window("Flight Reservation").WinButton("FLIGHT").Click
Window("Flight Reservation").Dialog("Flights Table").WinList("From").Select "19097 LON "
Window("Flight Reservation").Dialog("Flights Table").WinButton("OK").Click
```

with the following:

```
With Window("Flight Reservation")
    .WinComboBox("Fly From:").Select "London"
    .WinComboBox("Fly To:").Select "Los Angeles"
    .WinButton("FLIGHT").Click
    With .Dialog("Flights Table")
        .WinList("From").Select "19097 LON "
        .WinButton("OK").Click
    End With 'Dialog("Flights Table")
End With 'Window("Flight Reservation")
```

You can instruct UFT to automatically generate **With** statements when you record a test or to generate **With** statements for any existing action. You can also remove **With** statements from an action.

Using **With** statements in your test has no effect on the run session itself, only on the way your test appears in the Editor. Generating **With** statements for your test does not affect the Keyword View in any way.

For more details, see "How to Generate With Statements for Your Test" on page 978.

Test Synchronization

Relevant for: GUI tests and scripted GUI components

When you run a test, your application may not always respond with the same speed. For example, it might take a few seconds:

- for a progress bar to reach 100%
- for a status message to appear
- for a button to become enabled
- for a window or pop-up message to open

You can handle these anticipated timing problems by synchronizing your test to ensure that UFT waits until your application is ready before performing a certain step.

There are several options that you can use to synchronize your test:

- You can insert a **synchronization point**, which instructs UFT to pause the test until an object property achieves the value you specify. When you insert a synchronization point into your action, a **WaitProperty** statement is added to the action.
- You can insert **Exist** or **Wait** statements that instruct UFT to wait until an object exists or to wait

a specified amount of time before continuing the test.

- You can modify the default amount of time that UFT waits for a Web page to load.
- When working with tests, you can increase the default timeout settings for a test to instruct UFT to allow more time for objects to appear.

This section also includes:

Synchronization Points

Relevant for: GUI tests and scripted GUI components

If you do not want UFT to perform a step or checkpoint until an object in your application achieves a certain status, you should insert a synchronization point to instruct UFT to pause the test until the object property achieves the value you specify (or until a specified timeout is exceeded).

For example, suppose you record a test on a flight reservation application. You insert an order, and then you want to modify the order. When you click the **Insert Order** button, a progress bar is displayed and all other buttons are disabled until the progress bar reaches 100%. Once the progress bar reaches 100%, you record a click on the **Update Order** button.

Without a synchronization point, UFT may try to click the **Update Order** button too soon during a test run (if the progress bar takes longer than the test's object synchronization timeout), and the test will fail.

You can insert a synchronization point that instructs UFT to wait until the **Update Order** button's **enabled** property equals 1. For details, see "Add Synchronization Point Dialog Box" on page 982.

> **Tip:** UFT must be able to identify the specified object to perform a synchronization point. To instruct UFT to wait for an object to open or appear, use an **Exist** or **Wait** statement. For details, see "Exist and Wait Statements" below.

Exist and Wait Statements

Relevant for: GUI tests and scripted GUI components

You can enter **Exist** and/or **Wait** statements to instruct UFT to wait for a window to open or an object to appear. Exist statements return a boolean value indicating whether or not an object currently exists. Wait statements instruct UFT to wait a specified amount of time before proceeding to the next step. You can combine these statements within a loop to instruct UFT to wait until the object exists before continuing with the test.

For example, the following statements instruct UFT to wait up to 20 seconds for the Flights Table dialog box to open.

```
blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
counter=1
While Not blnDone
    Wait (2)
    blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
    counter=counter+1
    If counter=10 then
        blnDone=True
    End if
Wend
```

For more details on **While**, **Exist**, and **Wait** statements, see the *HP UFT Object Model Reference for GUI Testing*.

Timeout Values Modification

Relevant for: GUI tests and scripted GUI components

If you find that, in general, the amount of time UFT waits for objects to appear or for a browser to navigate to a specified page is insufficient, you can increase the default object synchronization timeout values for your test and the browser navigation timeout values for your test.

Alternatively, if you insert synchronization points and **Exist** and/or **Wait** statements for the specific areas in your test where you want UFT to wait a longer time for an event to occur, you may want to decrease the default timeouts for the rest of your test.

- When working with tests, to modify the maximum amount of time that UFT waits for an object to appear, change the **Object Synchronization Timeout** in the **File > Settings > Run** pane. For details, see "Run Pane (Test Settings Dialog Box)" on page 601.
- To modify the amount of time that UFT waits for a Web page to load, change the **Browser Navigation Timeout** in the **File > Settings > Web** pane. For details, see the *HP Unified Functional Testing Add-ins Guide*.

Message Statements

Relevant for: GUI tests and scripted GUI components

You can generate messages in your test that are displayed in the Run Results Viewer. You can also choose to display messages on screen while running your test.

This section includes:

- "Sending Messages to the Run Results" on the next page
- "Displaying Messages During the Run Session" on page 977

Sending Messages to the Run Results

You can send the following types of messages to the run results:

- **Messages specific to a run session.** You can add notes to the run results to provide additional information about the test or component run. For example, you can specify the application version tested, the operating system used, languages, and so on.

  In the Run Results Viewer, this information is displayed in the **Executive Summary** page. For details, see the Result Details Pane (described in the *HP Run Results Viewer User Guide*).

  You add a note by inserting a **Reporter.ReportNote** step in your test or component.

  Example:

  ```
  Reporter.ReportNote "This test was run from 12.34.56.89 using a wireless
   connection."
  ```

  For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

- **Messages specific to a step.** You can define a message that is sent to your run results. For example, suppose you want to check that a password edit box exists in the Mercury Tours site. If the edit box exists, then a password is entered. Otherwise, a message is sent to the run results indicating that the object is absent.

  You send messages to the run results using a **Reporter.ReportEvent** step.

  Example:

  ```
  Reporter.ReportEvent micFail, "Password edit box", "Password edit box does not
   exist"
  ```

  In this example, micFail indicates the status of the report (failed), Password edit box is the report name, and Password edit box does not exist is the report message.

  List of Supported Statuses:

  - **micPassed.** Causes this step to pass. Sends the specified message to the report.
  - **micFailed.** Causes this step (and therefore the test itself) to fail. Sends the specified message to the report.
  - **micDone.** Sends a message to the report without affecting the pass/fail status of the step.
  - **micWarning.** Sends a warning status for the step, but does not cause the test to stop running, and does not affect its pass/fail status.

Displaying Messages During the Run Session

In addition to sending messages to the Run Results, you can generate messages in the following ways:

- Use the **MessageBox** VBScript function in your test to display information during the run session. The run session pauses until the message box is closed. For more details, see the VBScript documentation from the **UFTHelp** menu (**Help > HP Unified Functional Testing Help > VBScript Reference**).

- Use the **Print** Utility statement in your test to display information in the Output pane while still continuing the run session. For example, the following example iterates all the items in the **Flight Table** dialog (in the sample Flight application) and uses the **Print** Utility statement to print the content of each item to the Output pane.

```
Set FlightsList = Window("Flight Reservation").Dialog("Flights Table").
    WinList("From")
For i = 1 to FlightsList.GetItemsCount
        Print FlightsList.GetItem(i - 1)
Next
```

For details on using the Output pane, see "Output Pane User Interface" on page 381.

Tasks

How to Insert Steps Using the Step Generator

Relevant for: GUI tests and scripted GUI components

This task describes how to insert steps using the Step Generator.

1. ### Prerequisites

   Select where in your test the new step should be inserted.

2. ### Open the Step Generator dialog box

   Open the Step Generator from one of the following locations:

   - Keyword View
   - Editor
   - Active Screen

   For details, see "Step Generator Dialog Box" on page 983.

3. ### Define the category, type and operations for the step

   First select the category for the step operation (test object, Utility object or function) and the required object or the function library source (for example, built-in or local script functions). You can then select the appropriate operation (method, property, or function) and define the arguments and return values, parameterizing them if required.

4. ### View the step documentation or syntax

   In the Step Generator, view the step documentation or statement syntax and add your new step or statement to your test or function library.

5. ### Results

   The Step Generator inserts a step with the correct syntax into your test. You can continue to add further steps at the same location without closing the Step Generator.

How to Generate With Statements for Your Test

Relevant for: GUI tests and scripted GUI components

This task describes the steps to generate and manage **With** statements in your test.

This task includes the following steps:

- "Instruct UFT to generate With statements while recording" below
- "Generate With statements for existing actions in the Editor" below
- "Remove With statements from an action in the Editor" on the next page

Instruct UFT to generate With statements while recording

1. In the **General** pane of the GUI Testing tab of the Options dialog box (**Tools > Options > GUI Testing** tab **> General** tab), select **Automatically generate "With" statements after recording** option. For details, see "General Pane (Options Dialog Box > GUI Testing Tab)" on page 536.

2. In the **Generate "With" statements for __ or more objects** box, enter the minimum number of consecutive, identical objects for which you want to apply the **With** statement. The default is 2.

   For example, if you only want to generate a **With** statement when you have three or more consecutive statements based on the same object, enter 3.

3. Begin recording your test. While recording, statements are recorded normally. When you stop recording, the statements in all actions recorded during the current recording session are automatically converted to the **With** format.

Generate With statements for existing actions in the Editor

You can instruct UFT to generate **With** statements for any action displayed in the Editor, and to enable the statement completion functionality within existing **With** statements.

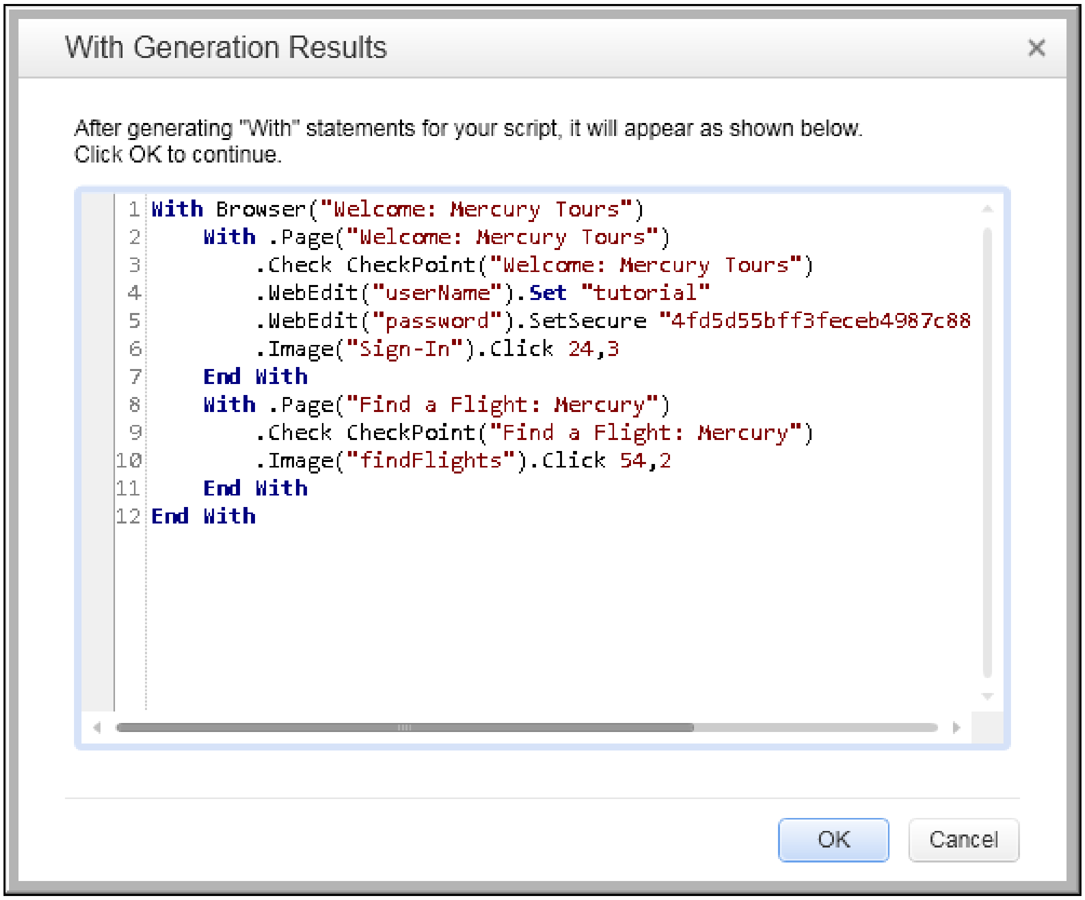
To generate With statements for existing actions:

1. In the **General** pane of the GUI Testing tab of the Options dialog box (**Tools > Options > GUI Testing** tab **> General** tab), select the **Automatically generate "With" statements after recording** option. For details, see "General Pane (Options Dialog Box > GUI Testing Tab)" on page 536.

2. Confirm that the proper number is set for the **Generate "With" statements for __ or more objects**. The default is 2.

3. Display the action for which you want to generate **With** statements.

4. From the Editor, select **Edit > Format > Apply "With" to Script**. The "With" Generation Results window opens.



With Generation Results

After generating "With" statements for your script, it will appear as shown below.
Click OK to continue.

```
With Browser("Welcome: Mercury Tours")
    With .Page("Welcome: Mercury Tours")
        .Check CheckPoint("Welcome: Mercury Tours")
        .WebEdit("userName").Set "tutorial"
        .WebEdit("password").SetSecure "4fd5d55bff3feceb4987c88
        .Image("Sign-In").Click 24,3
    End With
    With .Page("Find a Flight: Mercury")
        .Check CheckPoint("Find a Flight: Mercury")
        .Image("findFlights").Click 54,2
    End With
End With
```

Each **With** statement contains only one object.

To confirm the generated results, click **OK**. The **With** statements are applied to the action.

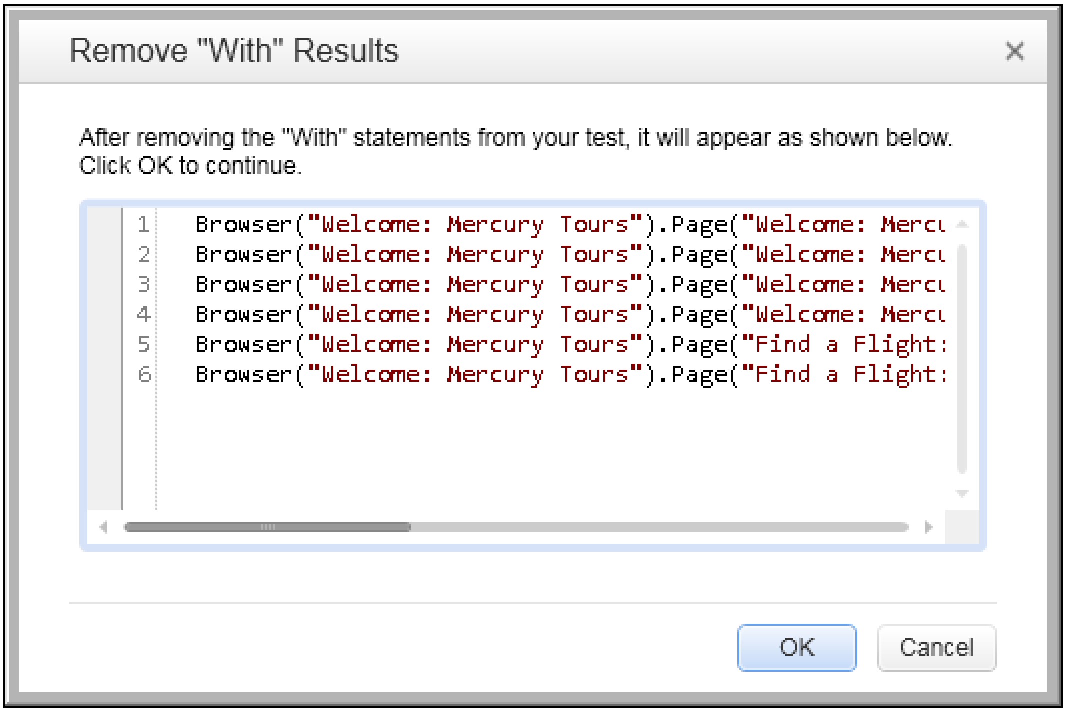
> **Tip:** If you type a **With** statement (as opposed to creating it using the procedure described above), select **Edit > Format > Apply "With" to Script** to enable statement completion within the **With** statement.

Remove With statements from an action in the Editor

You can remove all the **With** statements in an action displayed in the Editor.

To remove With statements from an action:

1. Display the action for which you want to remove **With** statements.
2. In the Editor, select **Edit > Format > Remove "With" Statements**. The Remove "With" Results window opens.



Remove "With" Results

After removing the "With" statements from your test, it will appear as shown below.
Click OK to continue.

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercu
Browser("Welcome: Mercury Tours").Page("Welcome: Mercu
Browser("Welcome: Mercury Tours").Page("Welcome: Mercu
Browser("Welcome: Mercury Tours").Page("Welcome: Mercu
Browser("Welcome: Mercury Tours").Page("Find a Flight:
Browser("Welcome: Mercury Tours").Page("Find a Flight:
```

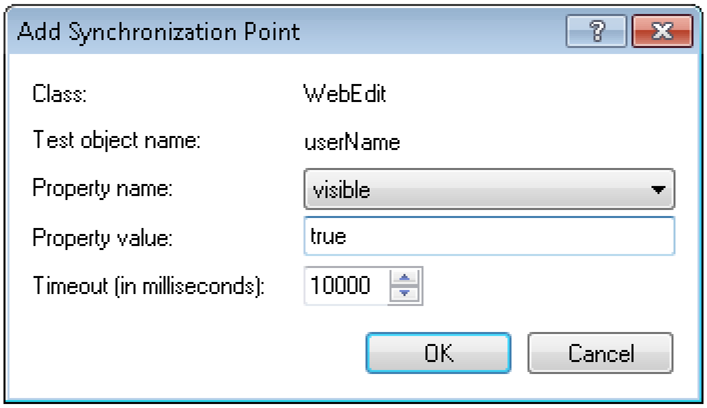
3. To confirm the results, click **OK**. The **With** statements are replaced with the standard statement format.

Reference

Add Synchronization Point Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to insert a **WaitProperty** statement to synchronize your test.



Class: WebEdit

Test object name: userName

Property name: visible

Property value: true

Timeout (in milliseconds): 10000

To access	1. Ensure that a GUI test, action, or component is in focus in the document pane. 2. Start a recording session. 3. Display the screen or page in your application that contains the object for which you want to insert a synchronization point. 4. In UFT, select **Design > Synchronization Point**.
Important information	• When you insert a synchronization point, the pointer changes to a pointing hand. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196. • For details on the **WaitProperty** method, see the *HP UFT Object Model Reference for GUI Testing*.
See also	"Test Synchronization" on page 973

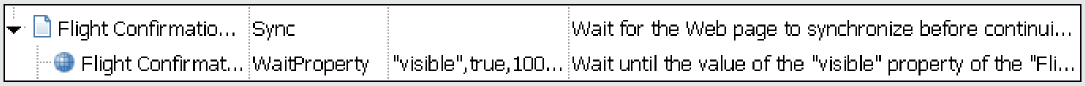
User interface elements are described below:

UI Elements	Description
Class	The test object class of the selected object.
Test object name	The name of the selected object.

UI Elements	Description
Property name	The list of the identification properties associated with the object class. Select the property you want to use for the synchronization point.
Property value	Enables you to specify the property value for which UFT should wait before continuing to the next step in the test. The property values that the object has at the time that you insert the synchronization point do not impact the specified synchronization point.
Timeout (in milliseconds)	The time (in milliseconds) that you want UFT to wait before continuing to the next step, if the specified property value was not achieved.

View Example

After you insert a synchronization point for the **Flight Confirmation** button, it may look something like this:



Flight Confirmatio...	Sync		Wait for the Web page to synchronize before continui...
Flight Confirmat...	WaitProperty	"visible",true,100...	Wait until the value of the "visible" property of the "Fli...

In the Editor, this is displayed as:

Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").Sync

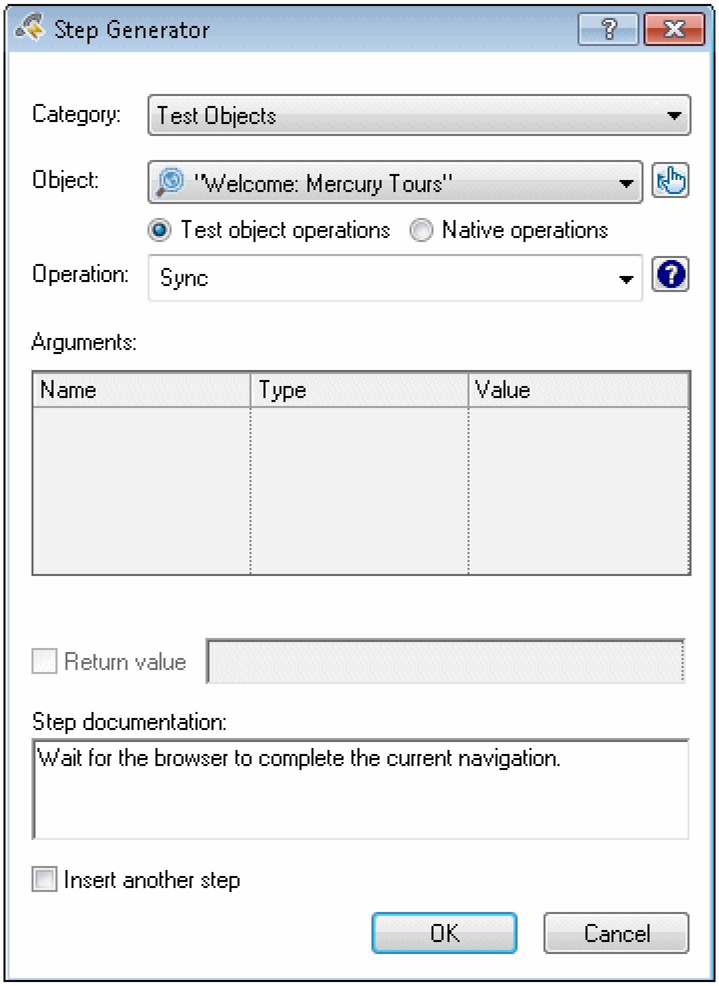
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").WebElement("Flight Confirmation #").WaitProperty "visible",

true, 10000

Step Generator Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to add steps that perform operations, using test object methods (for tests only), Utility object methods, or function calls.



Category: Test Objects

Object: "Welcome: Mercury Tours"

● Test object operations ○ Native operations

Operation: Sync

□ Return value

Step documentation:

Wait for the browser to complete the current navigation.

□ Insert another step

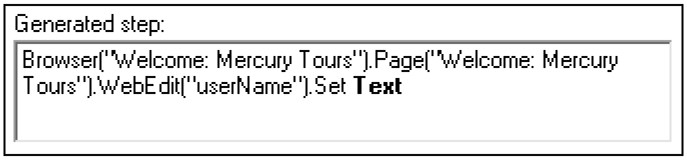
To access	This dialog box can be accessed from the following locations: • Keyword View • Editor • Function Library • Active Screen To open the dialog box, do the following: 1. Ensure that a GUI test or action is in focus in the document pane. 2. Select the location to insert the step, and do one of the following: ▪ Right-click and select **Insert Step > Step Generator**, or **Insert Step > Step Generator**. ▪ Press F7 (not available for Active Screen). ▪ Select **Design > Step Generator**.
Important information	• Although the Step Generator shows information regarding the currently selected step or function, selections that you make in the Step Generator add a new step or function to your test; they do not modify the existing step or function. • In the Step Generator, if you add an operation that returns an object, and the assignment in the test is missing a **Set** statement, the run session will fail.
Relevant tasks	"How to Insert Steps Using the Step Generator" on page 978

General User Interface Elements

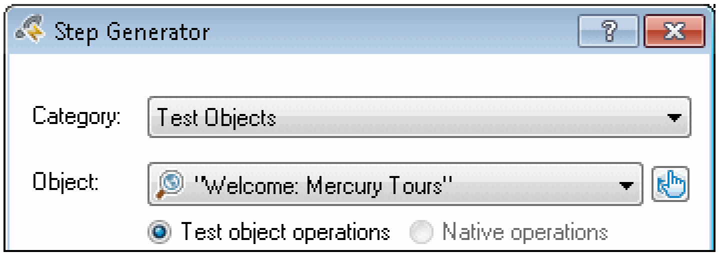
UI Elements	Description
Category	The type of step to add. The following options are available: • **Test Objects.** Enables you to select a test object and operation for the step (for tests only). For details, see "Test Object Category" on page 988. • **Utility Objects.** Enables you to select a Utility object and operation for the step. For details, see "Utility Object Category" on page 990. • **Functions.** Enables you to select a function for the step from the available library functions (tests only), VBScript functions, and internal script functions. For details, see "Function Category" on page 991.

UI Elements	Description
Object	The list of available objects. The list varies according to the type of object you select in the **Category** list box. **Note:** Available when the selected **Category** value is **Test Objects** or **Utility Objects**.
Library	The type of functions to display in the **Operation** list. Possible values: • **All.** Displays all functions relevant in the scope from which you opened the Step Generator. • **Library Functions.** Displays all functions defined in functions libraries associated with the action's test. (Available only when opening the Step Generator from an action.) • **Built-in functions.** Displays all of the functions that are part of the VBScript language. • **Local script functions.** Displays all of the functions defined in the action or function library from which you opened the Step Generator. **Note:** Available when the selected **Category** value is **Functions**.
Operation	**For objects:** The list of operations available for the selected object type in alphabetical order. **For functions:** The list of functions available from the selected library type in alphabetical order.
Arguments	The list of arguments for the operation, if applicable to the selected operation.
Arguments > Name	The name of the argument for the selected operation.
Arguments > Type	The type of the argument value for the selected operation.

UI Elements	Description
Arguments > Value	The value for the argument for the selected operation. Specify the argument value according to the following rules: - **Mandatory arguments.** If the name of the argument is followed by a red asterisk (*****), you must specify a value for the argument. You cannot insert the step or view the step documentation if the values have not been defined for all mandatory arguments. - **Optional arguments.** If the name of the argument is not followed by a red asterisk (*****), you can specify a value for the argument or leave the cell blank. If you do not specify a value, UFT uses the default value for the argument. (You can view the default value by moving the pointer over the cell). - **Required arguments.** If you specify a value for an optional argument, then you must also specify the values for any optional arguments that are listed before this argument. If you do not specify these values, UFT uses the default values for all required arguments. You can see the default value for each argument in a tooltip, by moving the pointer over the **Value** column. - **Parameterized arguments.** You can use a parameter for any argument value by clicking the parameterization button <#> . For details, see "Value Configuration Options Dialog Box" on page 1579. - **Predefined constants.** If an argument has a predefined list of values, UFT provides a drop-down list of possible values. If a list of values is provided, you cannot manually type a value in this box.
Return Value	The location for storing the return values of the operation, if applicable. For details on storing options, see "Storage Location Options Dialog Box" on page 991.

UI Elements	Description
Step documentation (Keyword View)	Summary information on the current step. The following options are available: • For the **Test Object** or **Utility Object** categories, the **Step documentation** box describes the operation performed by the step. When the step is inserted into your test, this description is displayed in the **Documentation** column in the Keyword View. **Note:** If any mandatory and required argument values have not been defined for the operation, the **Step documentation** box displays a warning message. • For **Functions** category, step documentation is available for user-defined functions, if you provided this information when defining them. For details, see "How to Create and Work with a User-Defined Function" on page 1048.
Generated step (Editor / Function Library)	The defined statement for the step. If all the mandatory and required argument values have not been defined for the operation, the names of the undefined arguments are highlighted in bold text. If you attempt to insert the step, an error message is displayed. **Example:**  Generated step: Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set **Text**
Insert another step	Enables you to insert the current step and continue adding steps at the same location. The **OK** button changes to **Insert**.

Test Object Category



Category: Test Objects

Object: "Welcome: Mercury Tours"

● Test object operations ○ Native operations

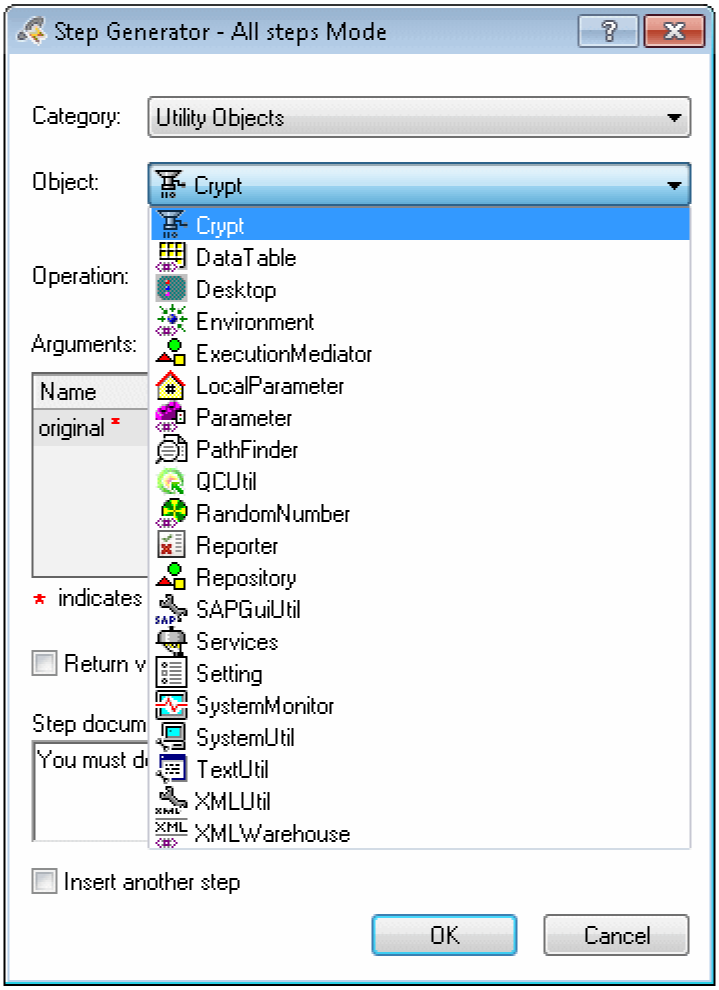
Important information	- You can select the object for the new step in the context of the currently selected step in your test. Alternatively, you can select any object from the object repository or from your application. - After you select the operation for your test object, you can define the relevant argument values. - If you click the **Operation Help** button when a native operation is selected, the *HP UFT Object Model Reference for GUI Testing* opens for the selected test object. For details on specific native operations, see the documentation for the environment or application you are testing.

User interface elements are described below:

UI Elements	Description
Object	All the objects in the object repository that are at the same hierarchical level and location as the currently selected step. **Note:** The objects are listed by name in alphabetical order.
👆	**Select Object**. Enables you to select an object from the object repository or from your application. For details, see "Select Test Object Dialog Box" on page 958.
Test object operations	The UFT operations that can be performed on a test object.
Native operations	The operations of the object in your application as defined by the object creator. **Note:** - If UFT cannot retrieve native operations for the selected object, the **Native operations** option is not available. - If you select a native operation, the Step Generator inserts a step using **.Object** syntax. For details on using the **Object** property, see "Native Properties and Operations" on page 1008.

Utility Object Category

This option enables you to specify a utility object to insert to your test.



Step Generator - All steps Mode

Category: Utility Objects

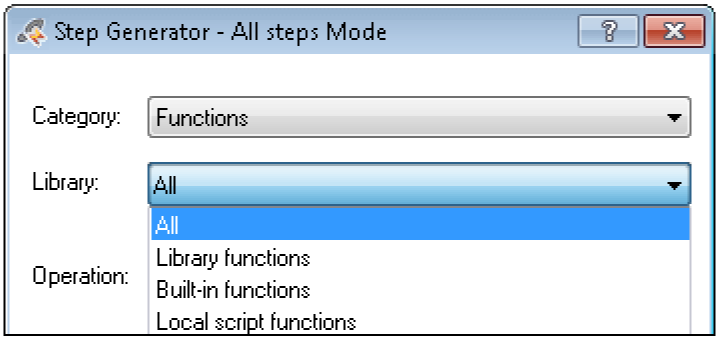
Object: Crypt

Important information	For details on Utility objects, see the Utility Objects section of the *HP UFT Object Model Reference for GUI Testing*.

User interface elements are described below:

UI Elements	Description
Object	The list of Utility objects that are available. **Note:** The image above displays the list of Utility objects that are available when you open the Step Generator from the Keyword View. When you open the Step Generator from the Editor, the list includes a number of additional Utility objects. If you have one or more add-ins loaded, the list may include additional Utility objects for those add-ins.

Function Category



Important information	For detailed information on a selected built-in VBScript function, see Microsoft's VBScript Reference or the *HP UFT Object Model Reference for GUI Testing*.

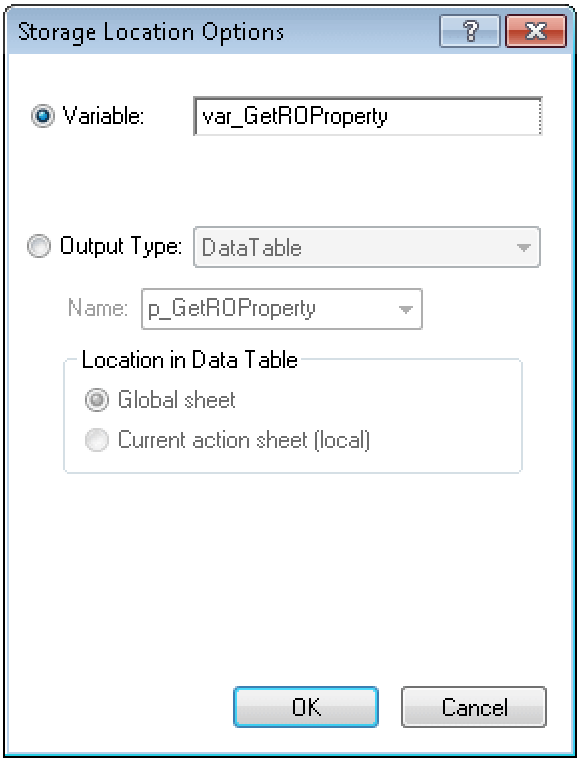
User interface elements are described below:

UI Elements	Description
Library	The list of function types. The following options are available: • **All.** Enables you to select a function from all the available functions and types. • **Library functions.** Enables you to select a function from any function library associated with your test (for tests only). For details on defining and using associated function libraries, see "Associated Function Libraries" on page 1031. • **Built-in functions.** Enables you to select any standard VBScript function supported by UFT. For details on working with VBScript, you can open the VBScript documentation from the UFT **Help** menu (**Help > HP Unified Functional Testing Help > VBScript Reference**). • **Local script functions.** Enables you to select any local function defined directly in the current action or function library.

Storage Location Options Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to specify how and where to store a return value for an operation that you have selected in the Step Generator dialog box. This dialog box also enables you to specify how and where to store the value for an output parameter for an action.



● Variable: var_GetROProperty

○ Output Type: DataTable

Name: p_GetROProperty

Location in Data Table

● Global sheet

○ Current action sheet (local)

To access	- In the "Step Generator Dialog Box", make sure the **Return value** check box is selected, click the displayed return value and then the output storage button . - In the "Action Call Properties Dialog Box" (described on page 895), select an output parameter in the Parameter Values tab and click the output storage button in the **Store in** column.
Relevant tasks	"How to Create or Modify an Output Value Step" on page 1494
See also	"Default Output Definitions (tests only)" on page 1491

User interface elements are described below:

UI Elements	Description
Variable	Stores the value in a run-time variable for the duration of the run session. You can accept the default name assigned to the variable (if any) or enter a different variable name.
Output Type	Stores the value in an output parameter of the specified type.

Default Output Definitions for Action Parameter Values

When you select **Output Type** or an output action parameter value for a nested action:

- If at least one output action parameter is defined in the action calling the nested action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the "Action Properties Dialog Box" (described on page 900) of the calling action.
- If no output action parameters are defined in the calling action, the default output type is **Data Table**, and UFT creates a new Data pane output name based on the selected value in the Global sheet of the Data pane.

When you select **Output Type** for an output action parameter value for a top-level action:

- If at least one output action parameter is defined in the test, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Test Properties dialog box.
- If no output action parameters are defined in the test, the default output type is **Data Table**, and UFT creates a new Data pane output name based on the selected value. The value is created in the Global sheet of the Data pane.

Chapter 36: Programming in GUI Testing Documents in the Editor

Relevant for: GUI tests, scripted GUI components, and function libraries

This chapter includes:

Concepts

Programming in GUI Testing Documents in the Editor - Overview

Relevant for: GUI actions, scripted GUI components, and function libraries

GUI test actions and scripted GUI components are composed of statements coded in the Microsoft VBScript programming language. The Editor provides an alternative to the Keyword View for testers who are familiar with VBScript. In the Editor, you can view an action or scripted component in VBScript.

When working with GUI tests, and both scripted and keyword GUI components, you can also create and work with function libraries in UFT using VBScript. This enables you to increase the power and flexibility of your tests and components.

If you are familiar with VBScript, you can add and update statements and enhance your tests, components, and function libraries with programming. This enables you to increase a test or component's power and flexibility.

This chapter provides a brief introduction to VBScript, and shows you how to enhance your actions, scripted components, and function libraries using a few simple programming techniques.

> **Note:** For details about using the text and code editor abilities available in the Editor, see "Editing Text and Code Documents" on page 305.

To learn about working with VBScript, you can view the VBScript documentation available from the UFT **Help** menu (**Help > HP Unified Functional Testing Help > VBScript Reference**).

You can add statements that perform operations on objects or retrieve information from your application. For example, you can add a step that checks that an object exists, or you can retrieve the return value of an operation.

You can add steps to your action, component, or function library either manually or using the Step Generator. For details on using the Step Generator, see "How to Insert Steps Using the Step Generator" on page 978.

Programmatic Descriptions

Relevant for: GUI actions, scripted GUI components, and function libraries

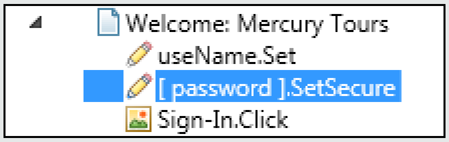
When UFT learns an object in your application, it adds the appropriate test object to the object repository. After the object exists in the object repository, you can add statements in the Editor to perform additional operations on that object. To add these statements, you usually enter the name (not case sensitive) of each of the objects in the object's hierarchy as the object description, and then add the appropriate operation.

Because each object in the object repository has a unique name, the object name is all you need to specify. During the run session, UFT finds the object in the object repository based on its name and parent objects, and uses the stored test object description for that test object to identify the object in your application.

You can also instruct UFT to perform operations on objects without referring to the object repository or to the object's name. To do this, you provide UFT with a list of properties and values that UFT can use to identify the object or objects on which you want to perform an operation. You can also describe an Insight test object, by providing a file containing an image of the control as a description property.

Such a **programmatic description** can be very useful if you want to perform an operation on an object that is not stored in the object repository. You can also use programmatic descriptions to perform the same operation on several objects with certain identical properties, or to perform an operation on an object whose properties match a description that you determine dynamically during the run session.

In the Run Results, square brackets around a test object name indicate that the test object was created dynamically during the run session using a programmatic description or the **ChildObjects** method.



Use-case scenario:

Suppose you are testing a Web site that generates a list of potential employers based on biographical information you provide, and offers to send your resume to the employer names you select from the list. You want your test to select all the employers displayed in the list, but when you design your test, you do not know how many check boxes will be displayed on the page, and you cannot, of course, know the exact object description of each check box. In this situation, you can use a programmatic description to instruct UFT to perform a Set "ON" method for all objects that fit the description: HTML TAG = input, TYPE = check box.

Programmatic Description Types

There are two types of programmatic descriptions:

- **Static.** You list the set of properties and values that describe the object directly in a VBScript statement. For details, see "Static Programmatic Descriptions" on the next page.
- **Dynamic.** You add a collection of properties and values to a Description object, and then enter the Description object name in the statement. For details, see "Dynamic Programmatic Descriptions " on page 1000.

Using the **Static** type to enter programmatic descriptions directly into your statements may be easier for basic object description needs. However, in most cases, using the **Dynamic** type provides more power, efficiency, and flexibility.

This section also includes:

Static Programmatic Descriptions

Relevant for: GUI actions, scripted GUI components, and function libraries

You can describe an object directly in a statement by specifying **property:=value** pairs describing the object instead of specifying an object's name.

The general syntax is:

*TestObject***(**"PropertyName1:=PropertyValue1", "..." ,
"PropertyNameX:=PropertyValueX"**)**

TestObject. The test object class.

PropertyName:=PropertyValue. The identification property and its value. Each **property:=value** pair should be separated by commas and quotation marks.

To describe an Insight test object, specify the **ImgSrc** property, with the **PropertyValue** providing the file system path or ALM path to an image of the control. (To specify an ALM path to a file located in the ALM Test Resources module, type: [QualityCenter\Resources] Subject\<folder and file name>).

The statement below specifies a WebEdit test object in the Mercury Tours page with the Name author and an index of 3. During the run session, UFT finds the WebEdit object with matching property values and enters the text Mark Twain.

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit
  ("Name:=Author", "Index:=3").Set "Mark Twain"
```

The statement below specifies an InsightObject test object in the Calculator window, with the image in the C:\AllMyFiles\Button6.bmp file. This file contains an image of the **6** button. During a run session, UFT finds the area on the calculator that looks like this image, and clicks its center.

```
Window("Calculator").InsightObject("ImgSrc:=C:\AllMyFiles\Button6.bmp").Click
```

For more details, see "Using Programmatic Descriptions for Insight Test Objects" on page 1000.

Guidelines for Using Static Programmatic Descriptions

When working with static programmatic descriptions, be aware of the following guidelines:

- "Regular Expressions in Programmatic Descriptions" below
- "Variables in Programmatic Descriptions" below
- "Programmatic Descriptions for Parent Test Objects" below
- "Reuse of Static Programmatic Descriptions" on the next page
- "Copying Programmatic Description Data from the Object Spy" on page 1000
- "Using Programmatic Descriptions for Insight Test Objects" on page 1000

Regular Expressions in Programmatic Descriptions

UFT evaluates all property values in programmatic descriptions as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *, ?, or +), use the \ (backslash) character to instruct UFT to treat the special characters as literal characters. For details on regular expressions related to tests and scripted components, see "Regular Expressions Overview" on page 318.

Variables in Programmatic Descriptions

You can enter a variable name as the property value if you want to find an object based on property values you retrieve during a run session. For example:

```
MyVar="some text string"
Browser("Hello").Page("Hello").Webtable("table").Webedit("name:=" & MyVar)
```

Programmatic Descriptions for Parent Test Objects

When using programmatic descriptions from a specific point within a test object hierarchy, you must continue to use programmatic descriptions from that point onward within the same statement. If you specify a test object by its object repository name after parent objects in the hierarchy have been specified using programmatic descriptions, UFT cannot identify the object.

Example:

- You can use the following statement, which uses object repository names for the parent objects and a programmatic description for the object on which the operation is performed:

```
Browser("Mercury Tours").Page("Mercury Tours").
 WebEdit("Name:=Author", "Index:=3").Set "Mark Twain"
```

- You can use the following statement since it uses programmatic descriptions throughout the entire test object hierarchy:

  ```
  Browser("Title:=Mercury Tours").Page("Title:=Mercury Tours").
   WebEdit("Name:=Author", "Index:=3").Set "Mark Twain"
  ```

- You can also use the statement below, since it uses programmatic descriptions from a certain point in the description (starting from the Page object description):

  ```
  Browser("Mercury Tours").Page("Title:=Mercury Tours").
   WebEdit("Name:=Author", "Index:=3").Set "Mark Twain"
  ```

- However, you cannot use the following statement, since it uses programmatic descriptions for the Browser and Page objects but then attempts to use an object repository name for the WebEdit test object:

  ```
  Browser("Title:=Mercury Tours").Page("Title:=Mercury Tours").
   WebEdit("Author").Set "Mark Twain"
  ```

In this case, UFT tries to locate the WebEdit object based on its name, but cannot locate it in the repository because the parent objects were specified using programmatic descriptions.

For details on working with test objects, see "Managing Test Objects in Object Repositories" on page 1198.

Reuse of Static Programmatic Descriptions

If you want to use the same static programmatic description several times in one action, scripted component, or function library, you may want to assign the object you create to a variable or use a **With** statement.

Example

Instead of entering:

```
Window("Text:=Myfile.txt - Notepad").Move 50,
Window("Text:=Myfile.txt - Notepad").WinEdit("AttachedText:=Find what:").Set "hello"
Window("Text:=Myfile.txt - Notepad").WinButton("Caption:=Find next").Click
```

You can enter:

```
Set MyWin = Window("Text:=Myfile.txt - Notepad")
MyWin.Move 50,
MyWin.WinEdit("AttachedText:=Find what:").Set "hello"
MyWin.WinButton("Caption:=Find next").Click
```

Alternatively, when working with actions or scripted components, you can use a **With** statement:

```
With Window("Text:=Myfile.txt - Notepad")
   .Move 50, 50
   .WinEdit("AttachedText:=Find what:").Set "hello"
   .WinButton("Caption:=Find next").Click
End With
```

For details on the **With** statement related to actions and scripted components, see "With Statement" on page 1026.

Copying Programmatic Description Data from the Object Spy

You can use the **Copy Identification Properties to Clipboard** option in the "Object Spy Dialog Box " (described on page 1189) to copy all identification properties and values for a selected object to the Windows Clipboard. The copied values are formatted in standard programmatic description syntax with line breaks between each property-value pair. For example:

```
"Class Name:=Image",
"abs_x:=585",
"abs_y:=573",
"alt:=Specials",
....
```

You can paste the copied data to any document and then copy selected lines (remove the line breaks) into a programmatic description. For details on this option, see "Object Spy Dialog Box " on page 1189.

Using Programmatic Descriptions for Insight Test Objects

When using programmatic descriptions for Insight test objects, consider the following:

- The file containing the image of the control must be a non-compressed image file (JPEG, GIF, BMP or PNG).
- When running the **Click** method on an Insight test object defined using a programmatic description, UFT clicks in the center of the control that matches the specified image.
- Make sure that the file specified in the programmatic description is accessible from any computer that runs the test or component.
- You can also include ordinal identifier properties in the programmatic description.

Dynamic Programmatic Descriptions

Relevant for: GUI actions, scripted GUI components, and function libraries

You can use the **Description** object to return a **Properties** collection object containing a set of **Property** objects. A **Property** object consists of a property name and value. You can then specify the returned **Properties** collection in place of an object name in a statement. (Each property object contains a property name and value pair.)

To create the **Properties** collection, you enter a **Description.Create** statement using the following syntax:

```
Set MyDescription = Description.Create()
```

After you have created a **Properties** object (such as MyDescription in the example above), you can enter statements to add, edit, remove, and retrieve properties and values to or from the **Properties** object during the run session. This enables you to determine which, and how many properties to include in the object description in a dynamic way during the run session.

After you fill the **Properties** collection with a set of **Property** objects (properties and values), you can specify the **Properties** object in place of an object name in a statement.

For example, instead of entering:

```
Window("Error").WinButton("text:=OK", "width:=50").Click
```

you can enter:

```
Set MyDescription = Description.Create()
MyDescription("text").Value = "OK"
MyDescription("width").Value = 50
Window("Error").WinButton(MyDescription).Click
```

When working with **Properties** objects, you can use variable names for the properties or values to generate the object description based on properties and values you retrieve during a run session.

You can create several **Properties** objects if you want to use programmatic descriptions for several objects.

For details on the **Description** and **Properties** objects and their associated methods, see the *HP UFT Object Model Reference for GUI Testing*.

Considerations for Description Objects

- By default, the value of all **Property** objects added to a **Properties** collection are treated as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *****, **?**, **+**), use the \ (backslash) character to instruct UFT to treat the special characters as literal characters. For details on regular expressions, see "Regular Expressions Overview" on page 318.

  You can set the **RegularExpression** property to False to specify a value as a literal value for a specific **Property** object in the collection. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

- When using programmatic descriptions from a specific point within a test object hierarchy, you must continue to use programmatic descriptions from that point onward within the same statement. If you specify a test object by its object repository name after other objects in the hierarchy have been described using programmatic descriptions, UFT cannot identify the object.

For example, you can use Browser(Desc1).Page(Desc1).Link(desc3), since it uses programmatic descriptions throughout the entire test object hierarchy.

You can also use Browser("Index").Page(Desc1).Link(desc3), since it uses programmatic descriptions from a certain point in the description (starting from the Page object description).

However, you cannot use Browser(Desc1).Page(Desc1).Link("Example1"), since it uses programmatic descriptions for the Browser and Page objects but then attempts to use an object repository name for the Link test object (UFT tries to locate the Link object based on its name, but cannot locate it in the repository because the parent objects were specified using programmatic descriptions).

Retrieving Child Objects

Relevant for: GUI actions, scripted GUI components, and function libraries

You can use the ChildObjects method to retrieve all objects located inside a specified parent object, or only those child objects that fit a certain programmatic description. To retrieve this subset of child objects, you first create a description object, and then you add the set of properties and values that you want your child object collection to match using the Description object.

> **Note:** You must use the Description object to create the programmatic description for the ChildObjects description argument. You cannot enter the programmatic description directly into the argument using the property:=value syntax.

After you build a description in your description object, use the following syntax to retrieve child objects that match the description:

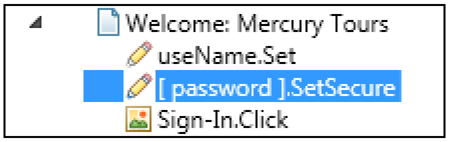
```
SetMySubSet=TestObject.ChildObjects(MyDescription)
```

Example

The statements below instruct UFT to select all of the check boxes on the Itinerary Web page:

```
Set MyDescription = Description.Create()
MyDescription("html tag").Value = "INPUT"
MyDescription("type").Value = "checkbox"
Set Checkboxes = Browser("Itinerary").Page("Itinerary").ChildObjects(MyDescription)
NoOfChildObjs = Checkboxes.Count
For Counter=0 to NoOfChildObjs-1
    Checkboxes(Counter).Set "ON"
Next
```

In the Run Results, square brackets around a test object name indicate that the test object was created dynamically during the run session using the ChildObjects method or a programmatic description.



Welcome: Mercury Tours
useName.Set
[password].SetSecure
Sign-In.Click

For details on the **ChildObjects** method, see the *HP UFT Object Model Reference for GUI Testing*.

Using the Index Property in Programmatic Descriptions

The index property can sometimes be a useful identification property for uniquely identifying an object. The index identification property identifies an object based on the order in which it appears within the source code, where the first occurrence is 0.

Index property values are object-specific. Thus, if you use an index value of 3 to describe a WebEdit test object, UFT searches for the fourth WebEdit object in the page.

If you use an index value of 3 to describe a WebElement object, however, UFT searches for the fourth Web object on the page regardless of the type, because the WebElement object applies to all Web objects.

For example, suppose you have a page with the following objects:

- An image with the name Apple
- An image with the name UserName
- A WebEdit object with the name UserName
- An image with the name Password
- A WebEdit object with the name Password

The description below refers to the third item in the list above, which is the first WebEdit object on the page with the name UserName:

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

The following description, however, refers to the second item in the list above, which is the first object of any type (WebElement) with the name UserName:

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

Note: If there is only one object, using index=0 does not retrieve it. You should not include the index property in the object description.

Programmatic Description Checks

Relevant for: GUI actions, scripted GUI components, and function libraries

You can compare the run-time value of a specified object property with the expected value of that property using either programmatic descriptions or user-defined functions.

Programmatic description checks are useful in cases in which you cannot apply a regular checkpoint, for example, if the object whose properties you want to check is not stored in an object repository. You can then write the results of the check to the Run Results report.

For example, suppose you want to check the run-time value of a Web button. You can use the **GetROProperty** or **Exist** operations to retrieve the run-time value of an object or to verify whether the object exists at that point in the run session.

Example

The following examples illustrate how to use programmatic descriptions to check whether the **Continue** Web button is disabled during a run session:

Using the **GetROProperty** operation:

```
ActualDisabledVal = Browser(micClass:="Browser").Page(micClass:="Page").WebButton
   (alt:="Continue").GetROProperty("disabled")
```

Using the **Exist** operation:

```
While Not Browser(micClass:="Browser").Page(micClass:="Page").WebButton
   (alt:="Continue").Exist(30)
Wend
```

By adding **Report.ReportEvent** statements, you can instruct UFT to send the results of a check to the run results:

```
If ActualDisabledVal = True Then
Reporter.ReportEvent micPass, "CheckContinueButton = PASS", "The Continue button
 is disabled, as expected."
Else
Reporter.ReportEvent micFail, "CheckContinueButton = FAIL", "The Continue button
 is enabled, even though it should be disabled."
```

You can also create and use user-defined functions to check whether your application is functioning as expected. The following example illustrates a function that checks whether an object is disabled and returns **True** if the object is disabled:

```
'@Description Checks whether the specified test object is disabled
'@Documentation Check whether the <Test object name> <test object type> is enabled.
Public Function VerifyDisabled (obj)
 Dim enable_property
 ' Get the disabled property from the test object
 enable_property = obj.GetROProperty("disabled")
 If enable_property = 1 Then ' The value is True (1)–the object is disabled
  Reporter.ReportEvent micPass, "VerifyDisabled Succeeded", "The test object is disabled, as ex
pected."
  VerifyDisabled = True
 Else
  Reporter.ReportEvent micFail, "VerifyDisabled Failed", "The test object is enabled, although it
should be disabled."
  VerifyDisabled = False
 End If
End Function
```

Note: For details on using the **GetROProperty** operation, see "Retrieving Native Properties" on page 1009. For details on using **While...Wend** statements, see "While...Wend Statement" on page 1026. For details on specific test objects, operations, and properties, see the *HP UFT Object Model Reference for GUI Testing*.

Opening and Closing Applications Programmatically

Relevant for: GUI actions and function libraries

In addition to using the "Record and Run Settings Dialog Box" (for tests) (described on page 865) to instruct UFT to open a new application when a run session begins, or manually opening the application you want to test, you can insert statements into your test or component that open and close the applications you want to test.

You can run any application from a specified location using a SystemUtil.Run statement. You can add these statements directly in your GUI test action or in a function library.

You can specify an application and pass any supported parameters, or you can specify a file name and the associated application starts with the specified file open.

This is especially useful in the following situations:

- **For tests:** If your test includes more than one application, and you selected the **Record and run test on any application** check box in the "Record and Run Settings Dialog Box" (described on page 865).
- **For components:** If you want to provide an operation (function) that opens an application from within a component.

You can close most applications using the Close method. You can also use SystemUtil statements to close applications.

For example, you could use the following statements to open a file named type.txt in the default text application (Notepad), type happy days, save the file using shortcut keys, and then close the application:

```
SystemUtil.Run "C:\type.txt", "","",""
Window("Text:=type.txt - Notepad").Type "happy days"
Window("Text:=type.txt - Notepad").Type micAltDwn & "F" & micAltUp
Window("Text:=type.txt - Notepad").Type micLShiftDwn & "S" & micLShiftUp
Window("Text:=type.txt - Notepad").Close
```

> **Note:**
>
> - When you specify an application to open using the "Record and Run Settings Dialog Box" (for tests only) (described on page 865), UFT does not add a SystemUtil.Run statement to your test.
> - The InvokeApplication method can open only executable files and is used primarily for backward compatibility.

For more details, see the *HP UFT Object Model Reference for GUI Testing*.

Comments, Control-Flow, and Other VBScript Statements

Relevant for: GUI tests and components

UFT enables you to incorporate decision-making into your test or component by adding conditional statements that control its logical flow. You can add the conditional statements directly in your action or component, or in the function libraries that they use.

In addition, you can define messages in your action or component that UFT sends to your run results.

To improve the readability of your actions and function libraries, you can also add comments to them.

For details on how to use these programming concepts in the Keyword View, see "Keyword View" on page 919.

> **Note:** The **VBScript Reference** (available from **Help > HP Unified Functional Testing Help**) contains Microsoft VBScript documentation, including VBScript, Script Runtime, and Windows Script Host.

For details, see:

- "Comments in VBScript" on page 1017
- "Calculations in VBScript" on page 1020
- "For...Next Statement" on page 1024
- "For...Each Statement" on page 1023
- "Do...Loop Statement" on page 1022
- "While...Wend Statement" on page 1026
- "If...Then...Else Statement" on page 1024
- "With Statement" on page 1026

Retrieving and Setting Identification Property Values

Relevant for: GUI tests and components

Identification properties are the set of properties defined by UFT for each object. You can set and retrieve a test object's identification property values, and you can retrieve the values of identification properties from a run-time object.

When you run your test or component, UFT creates a temporary version of the test object that is stored in the test object repository. You can use the GetTOProperty, GetTOProperties, and SetTOProperty methods in your action, component, or function library to set and retrieve the identification property values of the test object.

The GetTOProperty and GetTOProperties methods enable you to retrieve a specific property value or all the properties and values that UFT uses to identify an object.

The SetTOProperty method enables you to modify a property value that UFT uses to identify an object.

> **Note:** Because UFT refers to the temporary version of the test object during the run session, any changes you make using the SetTOProperty method apply only during the course of the run session, and do not affect the values stored in the test object repository.

For example, the following statements would set the **Submit** button's name value to my button, and then retrieve the value my button to the ButtonName variable:

```
Browser("QA Home Page").Page("QA Home Page").WebButton("Submit").SetTOProperty "Name", "my button"
ButtonName=Browser("QA Home Page").Page("QA Home Page").WebButton("Submit").GetTOProperty("Name")
```

You use the GetROProperty method to retrieve the current value of an identification property from a run-time object in your application.

For example, you can retrieve the target value of a link during the run session as follows:

```
link_href = Browser("HP Technologies").Page("HP Technologies").Link("Jobs").GetROProperty("href")
```

> **Tip:** If you do not know the identification properties of objects in your application, you can view them using the Object Spy. For details on the Object Spy, see "Object Spy Dialog Box " on page 1189.

For a list and description of identification properties supported by each object, and for more details on the GetROProperty, GetTOProperty, GetTOProperties, and SetTOProperty methods, see the *HP UFT Object Model Reference for GUI Testing*.

Native Properties and Operations

Relevant for: GUI tests and components

If the test object operations and identification properties available for a particular test object do not provide the functionality you need, you can access the native operations and properties of any run-time object in your application using the **Object** property.

You can view the native properties and their values, or the identification properties, of the run-time object associated with an object in your application, in the "Object Spy Dialog Box " (described on page 1189).

You can use the statement completion feature with object properties to view a list of the available native operations and properties of an object. For more details on the statement completion option, see "Statement Completion in the Editor" on page 306.

> **Tip:** If the object is a Web object, you can also reference its native properties in programmatic descriptions using the **attribute/property** notation. For details, see the *HP Unified Functional Testing Add-ins Guide*.

This section also includes:

- "Retrieving Native Properties" below
- "Activating Native Operations " below

Retrieving Native Properties

You can use the **Object** property to access the native properties of any run-time object. For example, you can retrieve the current value of the ActiveX calendar's internal **Day** property as follows:

```
Dim MyDay
Set MyDay=Browser("index").Page("Untitled").ActiveX("MSCAL.Calendar.7").Object.Day
```

For more details on the **Object** property, see the *HP UFT Object Model Reference for GUI Testing*.

Activating Native Operations

You can use the **Object** property to activate the internal operations of any run-time object. For example, you can activate the native **focus** method of the edit box as follows:

```
Dim MyWebEdit

Set MyWebEdit=Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Object

MyWebEdit.focus
```

For more details on the **Object** property, see the *HP UFT Object Model Reference for GUI Testing*.

Running DOS Commands

Relevant for: GUI actions, scripted GUI components, and function libraries

You can run standard DOS commands in your action, scripted component, or function using the VBScript Windows Scripting Host Shell object (WSCript.shell). For example, you can open a DOS command window, change the path to C:\, and run the **DIR** command using the following statements:

```
Dim oShell
Set oShell = CreateObject ("WSCript.shell")
oShell.run "cmd /K CD C:\ & Dir"
Set oShell = Nothing
```

For more details, see the Microsoft VBScript Language Reference (select **Help > HP Unified Functional Testing Help > VBScript Reference > VBScript**).

Filtering the GUI Steps Reported in the Run Session

Relevant for: GUI actions, scripted GUI components, and function libraries

You can use the **Report.Filter** method to determine which steps or types of steps are included in the Run Results. You can completely disable or enable reporting of steps following the statement, or you can indicate that you only want subsequent failed or failed and warning steps to be included in the report. You can also use the **Report.Filter** method to retrieve the current report mode.

For details, see "Report Modes" on page 1027.

Using the Windows API in GUI Testing

Relevant for: GUI actions, scripted GUI components, and function libraries

Using the Windows API, you can extend testing abilities and add usability and flexibility to your tests components. The Windows operating system provides a large number of functions to help you control and manage Windows operations. You can use these functions to obtain additional functionality.

The Windows API is documented in the Microsoft MSDN Web site, which can be found at: http://msdn.microsoft.com/en-us/library/ee663266

A reference to specific Windows API functions can be found at: http://msdn2.microsoft.com/en-us/library/Aa383749

For details on how to use the Windows API, see "How to Enhance Your Actions, Scripted Components, and Function Libraries Using the Windows API" on the next page.

Tasks

How to Enhance Your Actions, Scripted Components, and Function Libraries Using the Windows API

Relevant for: GUI actions, scripted GUI components, and function libraries

1. In MSDN, locate the function you want to use.

2. Read its documentation and understand all required parameters and return values.

3. Note the location of the Windows API function. Windows API functions are located inside Windows DLLs. The name of the .dll in which the requested function is located is usually identical to the Import Library section in the function's documentation. For example, if the documentation refers to User32.lib, the function is located in a .dll named User32.dll, typically located in your System32 library.

4. Use the UFT Extern object to declare an external function. For details, see the *HP UFT Object Model Reference for GUI Testing*.

   The following example declares a call to a function called **GetForegroundWindow**, located in user32.dll:

   ```
   extern.declare micHwnd, "GetForegroundWindow", "user32.dll", "GetForegroundWindow"
   ```

5. Call the declared function, passing any required arguments, for example:

   ```
   hwnd = extern.GetForegroundWindow()
   ```

   In this example, the foreground window's handle is retrieved. This can be useful if the foreground window is not in the object repository or cannot be determined beforehand (for example, a window with a dynamic title). You may want to use this handle as part of a programmatic description of the window, for example:

   ```
   Window("HWND:="&hWnd).Close
   ```

In some cases, you may have to use predefined constant values as function arguments. Since these constants are not defined in the context of your action, scripted component, or function, you need to find their numerical value to pass them to the called function. The numerical values of these constants are usually declared in the function's header file. A reference to header files can also be found in each function's documentation under the Header section. If you have Microsoft Visual Studio installed on your computer, you can typically find header files under X:\Program Files\Microsoft Visual Studio\VC98\Include.

For example, the GetWindow function expects to receive a numerical value that represents the relationship between the specified window and the window whose handle is to be retrieved. In the MSDN documentation, you can find the constants: GW_CHILD, GW_ENABLEDPOPUP, GW_HWNDFIRST, GW_HWNDLAST, GW_HWNDNEXT, GW_HWNDPREV and GW_HWNDPREV.

If you open the WINUSER.H file, mentioned in the **GetWindow** documentation, you will find the following flag values:

```
/*
 * GetWindow() Constants
 */
#define GW_HWNDFIRST   0
#define GW_HWNDLAST     1
#define GW_HWNDNEXT    2
#define GW_HWNDPREV    3
#define GW_OWNER       4
#define GW_CHILD       5
#define GW_ENABLEDPOPUP        6
#define GW_MAX  6
```

Example

The following example retrieves a specific menu item's value in the Notepad application:

```
' Constant Values:
const MF_BYPOSITION = 1024
' Windows API Functions Declarations
Extern.Declare micHwnd,"GetMenu","user32.dll","GetMenu",micHwnd
Extern.Declare micInteger,"GetMenuItemCount","user32.dll","GetMenuItemCount",micHwnd
Extern.Declare micHwnd,"GetSubMenu","user32.dll","GetSubMenu",micHwnd,micInteger
Extern.Declare micInteger,"GetMenuString","user32.dll","GetMenuString",micHwnd,micInteger,
micString+micByRef,micInteger,micInteger
' Notepad.exe
hwin = Window("Notepad").GetROProperty ("hwnd")' Get Window's handle
MsgBox hwin
' Use Windows API Functions
men_hwnd = Extern.GetMenu(hwin)' Get window's main menu's handle
MsgBox men_hwnd
item_cnt = Extern.GetMenuItemCount(men_hwnd)
MsgBox item_cnt
hSubm = Extern.GetSubMenu(men_hwnd,0)
MsgBox hSubm
rc = Extern.GetMenuString(hSubm,0,value,64,MF_BYPOSITION)
MsgBox value
```

Reference

Checkpoint and Output Statements

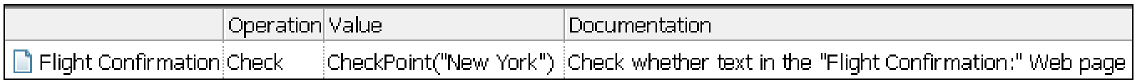
Relevant for: GUI actions and scripted GUI components

In UFT, you can create checkpoints and output values on pages, text strings, tables, and other objects. When you create a checkpoint or output value in the Keyword View, UFT creates a corresponding line in VBScript in the Editor. It uses the **Check** method to perform the checkpoint, and the **Output** method to perform the output value step.

For example, in the following statement UFT performs a check on the words New York:

```
Browser("Mercury Tours").Page("Flight Confirmation").Check Checkpoint("New York")
```

The corresponding step in the Keyword View is displayed as follows:



	Operation	Value	Documentation
Flight Confirmation	Check	CheckPoint("New York")	Check whether text in the "Flight Confirmation:" Web page

Note:

- The details about a checkpoint are set in the relevant Checkpoint Properties dialog box. The details about an output value step are set in the relevant Output Value Properties dialog box. The statement displayed in the Editor is a reference to the stored information. Therefore, you cannot insert a checkpoint or output value statement in the Editor manually.

- For details on inserting and modifying checkpoints, see "Checkpoints Overview" on page 1396. For details on inserting and modifying output values, see "Output Values Overview" on page 1488.

Basic VBScript Syntax

Relevant for: GUI actions, scripted GUI components, and function libraries

You use VBScript in UFT to develop actions, scripted components, or function libraries in the Editor that you can use in your GUI tests and components.

VBScript is an easy-to-learn, yet powerful scripting language. You can use VBScript to develop scripts to perform both simple and complex object-based tasks, even if you have no previous programming experience.

This section provides some basic guidelines to help you use VBScript statements to enhance your action, scripted component, or function library. For more detailed information on using VBScript, you can view the VBScript documentation from the **UFTHelp** menu (**Help > HP Unified Functional Testing Help > VBScript Reference**).

Each VBScript statement has its own specific syntax rules. If you do not follow these rules, errors will be generated when you run the problematic step.

Additionally, if you try to move to the Keyword View from the Editor, UFT lists any syntax errors found in the document in the Errors pane. You cannot switch to the Keyword View without fixing or eliminating the syntax errors. For details, see "Errors Pane User Interface" on page 374.

> **Tip:** You can check the syntax of your documents at any time by selecting **Design > Check Syntax**. In the Errors pane, you can view all of the syntax errors in the whole solution, or only the ones found in a selected test's actions and associated function libraries.

This section also includes:

General VBScript Syntax Rules and Guidelines

Relevant for: GUI actions, scripted GUI components, and function libraries

When working with actions, scripted components, or function libraries in the Editor, you should consider the following general VBScript syntax rules and guidelines:

- **Case-sensitivity.** By default, VBScript is not case sensitive and does not differentiate between upper-case and lower-case spelling of words, for example, in variables, object and operation names, or constants.

For example, the two statements below are identical in VBScript:

```
Browser("Mercury").Page("Find a Flight:").WebList("toDay").Select "31"
browser("mercury").page("find a flight:").weblist("today").select "31"
```

- **Text strings.** When you enter a value as a text string, you must add quotation marks before and after the string. For example, in the above segment of script, the names of the Web site, Web page, and edit box are all text strings surrounded by quotation marks.

  Note that the value 31 is also surrounded by quotation marks because it is a text string that represents a number and not a numeric value.

  In the following example, only the property name (first argument) is a text string and is in quotation marks. The second argument (the value of the property) is a variable and therefore does not have quotation marks. The third argument (specifying the timeout) is a numeric value, which also does not need quotation marks.

  ```
  Browser("Mercury").Page("Find a Flight:").WaitProperty("items count", Total_Items, 2000)
  ```

- **Variables.** You can specify variables to store strings, integers, arrays and objects. Using variables helps to make your script more readable and flexible. For details, see "Variables in VBScript" on page 1021.

- **Parentheses.** To achieve the desired result and to avoid errors, it is important that you use parentheses () correctly in your statements. For details, see "Parentheses in VBScript" on page 1019.

- **Indentation.** You can indent or outdent your script to reflect the logical structure and nesting of the statements. For details, see "Formatting VBScript Text" on the next page.

- **Comments.** You can add comments to your statements using an apostrophe ('), either at the beginning of a separate line, or at the end of a statement. It is recommended that you add comments wherever possible, to make your scripts easier to understand and maintain. For details, see "Formatting VBScript Text" on the next page, and "Comments in VBScript" on page 1017.

- **Spaces.** You can add extra blank spaces to your script to improve clarity. These spaces are ignored by VBScript.

For details on using specific VBScript statements to enhance your tests or components, see "Comments, Control-Flow, and Other VBScript Statements " on page 1006.

Handling VBScript Syntax Errors

Relevant for: GUI actions, scripted GUI components, and function libraries

The Errors pane lists the syntax errors found in your document, and enables you to locate each syntax error so that you can correct it. For details on the Errors pane, see "Errors Pane" on page 364.

You can view a description of each of the VBScript errors in the VBScript Reference. For details, select **Help > HP Unified Functional Testing Help > VBScript Reference > VBScript > Reference > Errors > VBScript Syntax Errors**.

> **Tip:** You can check the syntax of your documents at any time by selecting **Design > Check Syntax**. In the Errors pane, you can view all of the syntax errors in the whole solution, or only the ones found in a selected test's actions and associated function libraries.

For actions and scripted components: When you switch to the Keyword View from the Editor, UFT verifies the syntax. If a new or updated VBScript statement contains syntax errors, an error message is displayed in the Keyword View and the syntax error is displayed in the Errors pane. UFT is unable to display the document in the Keyword View until you have fixed all the syntax errors.

> **Tip:** The Microsoft VBScript Language Reference defines VBScript syntax errors as: "errors that result when the structure of one of your VBScript statements violates one or more of the grammatical rules of the VBScript scripting language."
>
> To learn about working with VBScript, you can view the VBScript Reference from the UFT**Help** menu (**Help > HP Unified Functional Testing Help > VBScript Reference**).

Formatting VBScript Text

Relevant for: GUI actions, scripted GUI components, and function libraries

When working with actions, scripted components, or function libraries in the Editor, it is important to follow accepted VBScript practices for comments and indentation.

Comments

Use comments to explain sections of an action, a scripted component, or a function library. This improves readability and makes your scripts easier to maintain and update. For details, see "Comments in VBScript" on the next page.

- **Adding Comments.** You can add comments to your statements by adding an apostrophe ('), either at the beginning of a separate line, or at the end of a statement.

  > **Tip:** You can comment a selected block of text by choosing **Edit > Format > Comment**.

> Each line in the block is preceded by an apostrophe.

- **Removing Comments.** You can remove comments from your statements by deleting the apostrophe ('), either at the beginning of a separate line, or at the end of a statement.

> **Tip:** You can remove the comments from a selected block or line of text by choosing **Edit > Format > Uncomment**.

Indentation

Use indentation to reflect the logical structure and nesting of your statements.

- **Indenting Statements.** You can indent your statements by selecting the text and choosing **Edit > Format > Indent**. If you want to indent multiple lines, you can also select the text and press the TAB key.

  The text is indented according to the **Tab spacing** selected in the **General** pane of the Text Editor tab in the Options dialog box (**Tools > Options > Text Editor** tab **> General** node). For details, see "General Pane (Options Dialog Box > Text Editor Tab)" on page 575.

  > **Note:** The **Indent selected text when using the Tab key** check box must be selected in the Editor Options dialog box, otherwise pressing the TAB key deletes the selected text.

- **Outdenting Statements.** You can outdent your statements by selecting **Edit > Format > Outdent** or by deleting the space at the beginning of the statements.

For more detailed information on formatting in VBScript, you can view the VBScript documentation from the **UFT Help** menu (**Help > HP Unified Functional Testing Help > VBScript Reference**).

Comments in VBScript

Relevant for: GUI actions, scripted GUI components, and function libraries

A comment is a line or part of a line in a script that is preceded by an apostrophe ('). During a run session, UFT does not process the comments. You can use comments to explain sections of an action, a scripted component, or a function library, to improve readability, and to make your scripts easier to maintain and update.

The following example shows how a comment describes the purpose of the statement below it:

```
`Sets the word "mercury" into the "username" edit box.
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set "mercury"
```

By default, comments are displayed in green inside your VBScript code. You can customize the appearance of comments in the **Fonts and Colors** pane of the **Text Editor** tab in the Options dialog box (**Tools > Options > Text Editor** tab **> General** node). For details, see "Fonts and Colors Pane (Options Dialog Box > Text Editor Tab)" on page 577.

Tips:

- You can comment a block of text by selecting **Edit > Format > Comment**.
- To remove the comment, select **Edit > Format > Uncomment**.

Note: You can also add a comment line using the VBScript **Rem** statement. For details, see the Microsoft VBScript Language Reference (select **Help > HP Unified Functional Testing Help > VBScript Reference > VBScript**).

Parameter Indications in VBScript

Relevant for: GUI actions and scripted GUI components

You can use UFT to enhance your tests by parameterizing values. A **parameter** is a variable that is assigned a value from an external data source or generator.

When you create a parameter in the Keyword View, UFT creates a corresponding line in VBScript in the Editor.

For example, if you define the value of a method argument as a Data pane parameter, UFT retrieves the value from the Data pane using the following syntax:

```
Object_Hierarchy.Method DataTable (parameterID, sheetID)
```

Item	Description
Object_ Hierarchy	The hierarchical definition of the test object, consisting of one or more objects separated by a dot.
Method	The name of the method that UFT executes on the parameterized object.
DataTable	The reserved object representing the data table.
parameterID	The name of the column in the data table from which to take the value.
sheetID	The name of the sheet in which the value is stored. If the parameter is a global parameter, dtGlobalSheet is the sheet ID.

Example

Suppose you are creating a test for the Mercury Tours site, and you select San Francisco as your destination. The following statement would be inserted into an action in your test in the Editor:

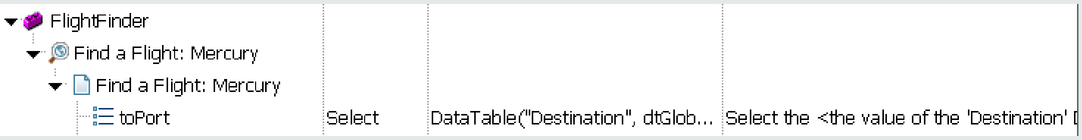
```
Browser("Welcome: Mercury").Page("Find a Flight:").WebList("toPort").Select "San Francisco"
```

Now suppose you parameterize the destination value, and you create a **Destination** column in the Data pane. The previous statement would be modified to the following:

```
Browser("Welcome: Mercury").Page("Find a Flight:").WebList("toPort").Select DataTable("Destin
ation",dtGlobalSheet)
```

In this example, Select is the method name, DataTable is the object that represents the data table, Destination is the Data pane parameter (column name), and dtGlobalSheet indicates the Global sheet in the Data pane.

In the Keyword View, this step is displayed as follows:



FlightFinder			
Find a Flight: Mercury			
Find a Flight: Mercury			
toPort	Select	DataTable("Destination", dtGlob...	Select the <the value of the 'Destination'

For more details on using and defining parameter values, see "Parameterizing Object Values" on page 1526.

Parentheses in VBScript

Relevant for: GUI actions, scripted GUI components, and function libraries

When programming in VBScript, it is important that you follow the rules for using or not using parentheses **()** in your statements.

You must use parentheses around method arguments if you are calling a method that returns a value and you are using the return value.

For example, use parentheses around method arguments if you are returning a value to a variable, if you are using the method in an **If** statement, or if you are using the **Call** keyword to call an action or function. When working with actions, you also need to add parentheses around the name of a checkpoint if you want to retrieve its return value.

Tip: If you receive an **Expected end of statement** error message when running a step, it may indicate that you need to add parentheses around the arguments of the step's method.

Example

Following are several examples showing when to use or not use parentheses.

The following example requires parentheses around the method arguments for the **ChildItem** method because it returns a value to a variable:

```
Set WebEditObj = Browser("Mercury Tours").Page("Method of Payment").WebTable("FirstName").C
hildItem (8, 2, "WebEdit", 0)
WebEditObj.Set "Example"
```

The following example requires parentheses around the method arguments because **Call** is being used:

```
Call RunAction("BookFlight", oneIteration)
```

or

```
Call MyFunction("Hello World")
...
...
```

The following example requires parentheses around the **WaitProperty** method arguments because the method is used in an **If** statement:

```
If Browser("index").Page("index").Link("All kinds of").WaitProperty("attribute/readyState", "complete", 4)

Then
    Browser("index").Page("index").Link("All kinds of").Click
End If
```

The following example requires parentheses around the **Check** method arguments, since it returns the value of the checkpoint:

```
a = Browser("MyBrowser").Page("MyPage").Check (CheckPoint("MyProperty"))
```

The following example does not require parentheses around the **Click** method arguments because it does not return a value:

```
Browser("Mercury Tours").Page("Method of Payment").WebTable("FirstName").Click 3,4
```

Calculations in VBScript

Relevant for: GUI actions, scripted GUI components, and function libraries

You can write statements that perform simple calculations using mathematical operators. For example, you can use a multiplication operator to multiply the values displayed in two text boxes in your Web site. VBScript supports the following mathematical operators:

Operator	Description
+	addition
–	subtraction
–	negation (a negative number)
*****	multiplication
/	division
^	exponent

In the following example, the multiplication operator is used to calculate the maximum luggage weight of the passengers at 100 pounds each:

```
'Retrieves the number of passengers from the edit box using the GetROProperty method
passenger = Browser ("Mercury_Tours").Page ("Find_Flights").WebEdit("numPassengers").GetR
OProperty("value")
'Multiplies the number of passengers by 100
weight = passenger * 100
'Inserts the maximum weight into a message box.
msgbox("The maximum weight for the party is "& weight &"pounds.")
```

Variables in VBScript

Relevant for: GUI actions, scripted GUI components, and function libraries

You can specify variables to store test objects or simple values in your action, scripted component, or function library. When using a variable for a test object, you can use the variable instead of the entire object hierarchy in other statements. Using variables in this way makes your statements easier to read and to maintain.

To specify a variable to store an object, use the **Set** statement, with the following syntax:

```
Set ObjectVar = ObjectHierarchy
```

In the example below, the **Set** statement specifies the variable **UserEditBox** to store the full **Browser.Page.WebEdit** object hierarchy for the **username** edit box. The **Set** method then enters the value John into the **username** edit box, using the **UserEditBox** variable:

```
Set UserEditBox = Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username")
UserEditBox.Set "John"
```

> **Note:** Do not use the **Set** statement to specify a variable containing a simple value (such as a string or a number). The example below shows how to define a variable for a simple value:
>
> ```
> MyVar = Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").GetTOProperty(
> "type")
> ```

You can also use the **Dim** statement to declare variables of other types, including strings, integers, and arrays. This statement is not mandatory, but you can use it to improve the structure of your action, scripted component, or function library.

The examples below demonstrate using the **Dim** statement to declare a variable:

In an action or scripted component (using the passengers variable):

```
Dim passengers
passengers = Browser("Mercury Tours").Page("Find Flights").WebEdit
```

```
("numpassengers").GetROProperty("value")
```

In a function libraries (using the actual_value variable):

```
Dim actual_value
    ' Get the actual property value
    actual_value = obj.GetROProperty(PropertyName)
```

These variables can then be used in different statements within the action, scripted component, or function library.

Do...Loop Statement

Relevant for: GUI actions, scripted GUI components, and function libraries

The **Do...Loop** statement instructs UFT to perform a statement or series of statements while a condition is true or until a condition becomes true. It has the following syntax:

```
Do [{while} {until} condition]
    statement
Loop
```

Item	Description
condition	A condition to be fulfilled.
statement	A statement or series of statements to be performed during the loop.

In the following example, UFT calculates the factorial value of the number of passengers using the **Do...Loop**:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
  WebEdit("numPassengers").GetROProperty("value")
total = 1
i = 1
Dowhile i <= passengers
   total = total * i
   i = i + 1
Loop
MsgBox "!" & passengers & "=" & total
```

For...Each Statement

Relevant for: GUI actions, scripted GUI components, and function libraries

A **For...Each** loop instructs UFT to perform one or more statements for each element in an array or an object collection. It has the following syntax:

```
For Each item In array
    statement
Next
```

Item	Description
item	A variable representing the element in the array.
array	The name of the array.
statement	A statement, or series of statements, to be performed during the loop.

The following example uses a **For...Each** loop to display each of the values in an array:

```
MyArray = Array("one","two","three","four","five")
For Each element In MyArray
  msgbox element
Next
```

Note: During a run session, if a **For Each** statement iterates on the **ParameterDefinitions** collection, the run may fail if the collection was retrieved directly before using the **For Each** statement. To prevent this, use other VBScript loop statements, such as **For** or **While**.

For...Next Statement

Relevant for: GUI actions, scripted GUI components, and function libraries

A **For...Next** loop instructs UFT to perform one or more statements a specified number of times. It has the following syntax:

```
For counter = start to end [Step step]
    statement
Next
```

Item	Description
counter	The variable used as a counter for the number of iterations.
start	The start number of the counter.
end	The last number of the counter.
step	The number to increment at the end of each loop. **Default = 1**. **Optional**.
statement	A statement, or series of statements, to be performed during the loop.

In the following example, UFT calculates the factorial value of the number of passengers using the **For** statement:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
  WebEdit("numPassengers").GetROProperty("value")
total = 1
For i=1 To passengers
  total = total * i
Next
MsgBox "!" & passengers & "=" & total
```

If...Then...Else Statement

Relevant for: GUI actions, scripted GUI components, and function libraries

The **If...Then...Else** statement instructs UFT to perform a statement or a series of statements based on specified conditions. If a condition is not fulfilled, the next **Elseif** condition or **Else** statement is examined. It has the following syntax:

```
If condition Then
    statement
```

```
ElseIf condition2 Then
    statement
Else
    statement
End If
```

Item	Description
condition	Condition to be fulfilled.
statement	Statement to be perform.

Example

In the following example, if the number of passengers is fewer than four, UFT closes the browser:

```
passengers = Browser("Mercury Tours").Page("Find Flights").WebEdit("numpassengers").GetRO
Property("value")
If (passengers < 4) Then
    Browser("Mercury Tours").Close
Else
    Browser("Mercury Tours").Page("Find Flights").Image("continue").Click 69,5
End If
```

The following example uses **If**, **ElseIf**, and **Else** statements to check whether a value is equal to 1, 2, or a different value:

```
value = 2
If value = 1 Then
  msgbox "one"
ElseIf value = 2 Then
  msgbox "two"
Else
  msgbox "not one or two"
End If
```

While...Wend Statement

Relevant for: GUI actions, scripted GUI components, and function libraries

A **While...Wend** statement instructs UFT to perform a statement or series of statements while a condition is true. It has the following syntax:

```
While condition
    statement
Wend
```

Item	Description
condition	A condition to be fulfilled.
statement	A statement or series of statements to be executed during the loop.

In the following example, UFT performs a loop using the **While** statement while the number of passengers is fewer than ten. Within each loop, UFT increments the number of passengers by one:

```
passengers = Browser("Mercury Tours").Page("Find Flights").WebEdit("numpassengers").GetRO
Property("value")
While passengers < 10
    passengers = passengers + 1
Wend
msgbox("The number of passengers in the party is " & passengers)
```

With Statement

Relevant for: GUI actions and scripted GUI components

With statements make your script more concise and easier to read and write or edit by grouping consecutive statements with the same parent hierarchy.

In addition, using **With** statements might help your script run faster. When running a **With** statement, UFT identifies the object in the application before running the first statement, but does not re-identify it before running each statement.

On the other hand, this can affect the running of your test if the object referenced by the **With** statement is refreshed, redrawn, or changed in some way in the application while running the **With** statement. To instruct UFT to re-identify the object in the application before running the next statement, add a statement that calls the **RefreshObject** test object operation. For details on the **RefreshObject** operation, see the *HP UFT Object Model Reference for GUI Testing*.

The **With** statement has the following syntax:

```
With object
```

```
    statements
End With
```

Item	Description
object	An object or a function that returns an object.
statements	One or more statements to be performed on an object.

Example

You could replace this script:

```
Window("Flight Reservation").WinComboBox("Fly From:").Select "London"
Window("Flight Reservation").WinComboBox("Fly To:").Select "Los Angeles"
Window("Flight Reservation").WinButton("FLIGHT").Click
Window("Flight Reservation").Dialog("Flights Table").WinList("From").
    Select "19097 LON"
Window("Flight Reservation").Dialog("Flights Table").WinButton("OK").Click
```

with the following:

```
With Window("Flight Reservation")
    .WinComboBox("Fly From:").Select "London"
    .WinComboBox("Fly To:").Select "Los Angeles"
    .WinButton("FLIGHT").Click
    With .Dialog("Flights Table")
        .WinList("From").Select "19097 LON"
        .WinButton("OK").Click
    End With 'Dialog("Flights Table")
End With 'Window("Flight Reservation")
```

Note that entering With statements in the Editor does not affect the Keyword View in any way.

Note: In addition to entering With statements manually, you can also instruct UFT to automatically generate With statements as you record or to generate With statements for an existing test. For more details, see "How to Generate With Statements for Your Test" on page 978.

Report Modes

Relevant for: GUI actions, scripted GUI components, and function libraries

For details on using report modes, see "Filtering the GUI Steps Reported in the Run Session" on page 1010.

The following report modes are available:

Mode	Description
0 or **rfEnableAll**	All events are displayed in the Run Results. **Default.**
1 or **rfEnableErrorsAndWarnings**	Only events with a warning or fail status are displayed in the Run Results.
2 or **rfEnableErrorsOnly**	Only events with a fail status are displayed in the Run Results.
3 or **rfDisableAll**	No events are displayed in the Run Results.

- To disable reporting of subsequent steps, enter the following statement:

```
Reporter.Filter = rfDisableAll
```

- To re-enable reporting of subsequent steps, enter:

```
Reporter.Filter = rfEnableAll
```

- To instruct UFT to include only subsequent failed steps in the Run Results, enter:

```
Reporter.Filter = rfEnableErrorsOnly
```

- To instruct UFT to include only subsequent failed or warning steps in the Run Results, enter:

```
Reporter.Filter = rfEnableErrorsAndWarnings
```

- To retrieve the current report mode, enter:

```
MyVar=Reporter.Filter
```

For more details, see the *HP UFT Object Model Reference for GUI Testing*.

Chapter 37: User-Defined Functions and Function Libraries

Relevant for GUI tests and components

> **Note:** The terms function, method, and operation are used interchangeably in this chapter.

This chapter includes:

Concepts

Function Library Overview

Relevant for: GUI tests and components

In addition to the test objects, methods, and built-in functions supported by the UFT Test Object Model, you can define your own function libraries containing VBScript functions, subroutines, statement, and so on, and then call their functions from your test or use their functions as operations in your test or component.

A **function library** is a separate document that contains Visual Basic script. Any text file written in standard VBScript syntax can be used as a function library.

Your function libraries can contain:

- **Function definitions (function signature and code).** You can call these functions from other functions, from actions in your test, or from your component. To call a function from a test or component, you must first associate the function library with the test or with the component's application area.

- **VBScript statements.** These are statements that are not contained within function definitions (for example, RegisterUserFunc statements). UFT runs all of these statements when it loads the function library.

Loading Function Libraries

At the beginning of a run session, UFT loads all of the function libraries associated with your test or with your component's application area.

In addition, you can use the **LoadFunctionLibrary** statement to dynamically load a function library during a run session.

Editing Function Libraries

When you edit function libraries you can do any of the following:

- Open and work on one or several function libraries at the same time, as each function library opens in a separate document tab.

- Use UFT to modify and debug any existing function libraries (.vbs, .txt, or .qfl files), even if they were created using an external editor.

  For more details:

  - On designing steps in function libraries, see "Programming in GUI Testing Documents in the Editor" on page 994.

  - On editing tools that UFT provides, such as syntax checking and statement completion, see "UFT File Operations" on page 127.

    - On using VBScript, see "Handling VBScript Syntax Errors" on page 1016 and "Basic VBScript Syntax" on page 1013.
    - On debugging function libraries, see "Debugging Tests and Components" on page 674.

- Choose to open a function library in edit mode or read-only mode:

    - **Edit mode.** Enables you to view and modify the function library. While the function library is open on your computer, other users can view the file in read-only mode, but they cannot modify it.
    - **Read-only mode.** Enables you to view the function library but not modify it. By default, when you open a function library that is currently open on another computer, it opens in read-only mode. You can also choose to open a function library in read-only mode if you want to review it, but you do not want to prevent another user from modifying it.

- Navigate directly from a function call in your document to its function definition in the source document. The function definition can be located either in the same document (action or function library) or in another function library that is associated with your test or component.

  If the document containing the function definition is already open, UFT activates the document (brings it into focus). If the document is closed, UFT opens it in read-only mode. For task details, see "Navigate to the function's definition - optional" on page 1051.

- Close a function library after you finish editing it, leaving your UFT session open.

- Save the function library to your ALM project or to the file system. By default, UFT saves a function library with a .qfl extension, unless you specify a different extension, such as .vbs or .txt.

  For tests: You can also save a function library as an attachment to a test in ALM for storage purposes only. To insert function calls from this function library into a test, you must first associate the function library with the test. For details, see "How to Manage Function Library Associations" on page 1045.

For task details, see "How to Create and Manage Function Libraries" on page 1041.

Associated Function Libraries

Relevant for: GUI tests and components

After you create your function libraries, you can associate them with your test or application area. This enables you to insert a call to a public function or subroutine in the associated function library from that test or any component associated with that application area.

At the beginning of a run session, UFT loads the function libraries associated with the test or application area, and can then access their functions during the run session. Public functions stored in function libraries can be called from any associated test or component, whereas private functions can be called only from within the same function library.

The order in the list of associated function libraries determines the order in which UFT searches for a function or subroutine that is called from a step in your test or component. If there are two functions or subroutines with the same name, UFT uses the first one it finds.

You can:

- Edit the list of associated function libraries for an existing test or application area
- Specify default function libraries for all new tests (tests only)

For details, see "How to Manage Function Library Associations" on page 1045.

To use a function library without associating it to your test or application area, you can load the function library dynamically during a run session using the **LoadFunctionLibrary** statement. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

If you dynamically load a function library during a run session, and a later step calls a function that has the same name as a function in an associated function library, the function in the dynamically loaded function library is used.

Working with Associated Function Libraries in ALM

Relevant for: GUI tests and components

You can associate a function library with a test, regardless of whether the function library is stored in the file system or your ALM project. However, if you are planning to use the function library in a business process test or if you are working with the Resources and Dependencies model, you must save it in your ALM project.

When working with ALM and associated function libraries, you must save the function library in the Test Resources module in your ALM project before you can associate it with the test or application area. You can add a new or existing function library to your ALM project.

If you add an existing function library from the file system to an ALM project, you are actually adding a copy of that file to the project. Therefore, if you later make modifications to either of these function libraries (in the file system or in your ALM project), the other function library remains unaffected.

A component accesses the functions that are associated with its application area. Therefore, any changes you make to a function library that is stored in your ALM project and associated with an application area may affect its associated components. When making changes to a function library that is stored in your ALM project and associated with an application area, consider the effect of the changes on the components that use this application area. In ALM, you can view the list of components using the application area in the Dependencies tab of the Business Components module. For details, see "Dependencies Tab (ALM Modules) " on page 762"Dependencies Tab (ALM Modules) " on page 762.

User-Defined Functions

Relevant for: GUI tests and components

For tests or scripted components, if you have segments of code that you need to use several times in your tests, you may want to create a user-defined function.

For keyword components, you can create user-defined functions to provide additional functionality.

You create the functions in VBScript. For details on using VBScript, see "Handling VBScript Syntax Errors" on page 1016 and "Basic VBScript Syntax" on page 1013.

A user-defined function encapsulates an activity (or a group of steps that require programming) into a keyword (also called an operation). By using user-defined functions, your tests or components are shorter, and easier to design, read, and maintain. You or a Subject Matter Expert can then call user-defined functions from an action or a component by inserting the relevant keywords (or operations) into that action or component.

For tests: Before you can call a function from an action, you must add the function definition to the action or to a function library that is associated with the action's test.

For components: Before you can call a function from a component, you must add the function definition to a function library that is associated with the component's application area.

Global Functions

A user-defined function is automatically defined as a global function. You can call global functions by typing them in the step or selecting them from the lists displayed in the following locations:

- The **Operation** box in the Step Generator, when the **Functions** category is selected (for function libraries)
- The **Operation** column in the Keyword View, when the **Operation** item is selected from the **Item** list
- The Editor, when using the statement completion feature

Functions That Are Registered to Test Objects

You can also register a user-defined function as a method for a UFT test object class (type). A registered method can either override the functionality of an existing test object method for the duration of a run session, or be registered as a new method for a test object class. You can call the test object method by typing it in the step or selecting it from the list of operations available for the test object.

For more details, see "Registered User-Defined Functions" on page 1035 and "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1051.

User-Defined Function Names

When deciding the name for your function, consider the following:

- During run-time, UFT searches the function libraries for the specified function in the order in which they are listed in the Solution Explorer. This order determines the function library priority.

  For tests, UFT searches for the specified function in the action **before** searching the function libraries.

  If UFT finds more than one function that matches the function name in a specific action or function library, it uses the last function it finds in that action or function library.

If UFT finds two functions with the same name in two different function libraries, it uses the function from the function library that has the higher priority. To avoid confusion, it is recommended that you verify that within the resources associated with a test or application area, each function has a unique name.

- When you create a user-defined function, do not give it the same name as a built-in function (for example, **GetLastError**, **MsgBox**, or **Print**). Similarly, do not use VBScript registered words (for example, **cStr**, **F1**, **ESC**) for function names. Built-in functions take priority over user-defined functions, so if you call a user-defined function that has the same name as a built-in function, the built-in function is called instead. For a list of built-in functions, see the **Built-in functions** list in the Step Generator (**Design > Step Generator**).

  For details on naming conventions for function names, see "Troubleshooting - Naming Conventions" on page 2269.

User-Defined Function Storage and Access

Relevant for: GUI tests and components

Using UFT, you can define and store your user-defined functions either in a function library (saved as a .qfl file, by default) or directly in an action within a test.

When you store a public function in a function library and associate the function library with a test or application area, the test or any component associated with that application area can call the public functions in that function library. For details, see "Associated Function Libraries" on page 1031.

You can access the functions that are stored in an associated function library from the Step Generator (for function libraries), the **Operation** column in the Keyword View, or the Toolbox pane, or you can enter these functions manually in a function library or an action.

You can also define private functions and store them in a function library. **Private functions** are functions that can be called only by other functions within the same function library. This is useful if you need to reuse segments of code in your public functions.

By implementing user-defined functions in function libraries and associating them with your test or component (via the component's application area), you and other users can choose functions that perform complex operations, such as adding if/then statements and loops, or working with utility objects—without adding the code directly to the test or needing any programming knowledge. In addition, you save time and resources by implementing and using reusable functions.

For tests:

- When you store a function in an action, it can be called only from within that action—the function cannot be called from any other action or test. This is useful if you do not want the function to be available outside of a specific action.

- If the same function name exists locally within your action and within an associated function library, the function defined in the action in used.

Registered User-Defined Functions

Relevant for: GUI tests and components

You can register a public user-defined function to a test object to instruct UFT to use your user-defined function as a method of a specified test object class for the duration of a test or component run, or until you unregister the method.

A registered method applies only to the run session in which you register it. All function registrations are cleared at the beginning of each run session.

When you register a function to a test object class, you can register the function as a new operation for the test object class, or you can choose to override the functionality of an existing operation. You can unregister the function to disable new operations or to return existing operations to their original UFT behavior.

For tests: If you call an external action that registers a method (and does not unregister it at the end of the action), the method registration remains in effect for the remainder of the test that called the action.

Availability of Registered User-Defined Functions (for tests)

After you register a function to a test object class, it can be called as a method of that test object class, in addition to being available as a global function.

UFT displays the function in the general **Operation** list in the Step Generator and in the list of operations available for the test object displayed in the following locations:

- The **Operation** box in the Step Generator, when a test object of the relevant class is selected.
- The **Operation** column in the Keyword View, when a test object of the relevant class is selected from the **Item** list.
- The Editor, when you type the name of a test object of the relevant class and use the statement completion feature.

When you register a function to a test object class, you can optionally define it as the default operation for that test object class. This instructs UFT to use the function as the test object operation by default, in the following situations:

- In the **Operation** column in the Keyword View, when you choose a test object of the relevant class in the **Item** list.
- In the **Operation** box in the Step Generator, when you choose a test object of the relevant class from the **Object** list.
- In the Editor, when you drag in a test object of the relevant class from the object repository.

Availability of Registered User-Defined Functions (for components)

After you register a function to a test object class, it can be called as a method of that test object class, in addition to being available as a global function.

UFT therefore displays the function in the Keyword View **Operation** list when a test object of that class is selected from the **Item** list, as well as in the general **Operation** list in the Step Generator (for function libraries).

When you register a function to a test object class, you can optionally define it as the default operation for that test object class. This instructs UFT to display the function in the **Operation** column in the Keyword View, by default, when you or a Subject Matter Expert choose a test object from the relevant class in the **Item** list.

This section also includes:

Preparing the User-Defined Function for Registration

Relevant for: GUI tests and components

When you run a statement containing a registered method, UFT sends the test object as the first argument. For this reason, your user-defined function must have at least one argument. Your user-defined function can have any number of arguments, or it can have only the test object argument.

If you register a user-defined function to override an existing test object method, then after the test object argument, the function must have the same number of arguments as the method it overrides.

> **Tip:** You can use the **parent** identification property to retrieve the parent of the object represented by the first argument in your function. For example:
> ParentObj = obj.GetROProperty("parent")

Registering User-Defined Functions as Test Object Methods

Relevant for: GUI tests and components

To register a user-defined function as a test object method, you enter a **RegisterUserFunc** statement in an action or function library. The **RegisterUserFunc** statement specifies the test object class, the name of your function, and the name of the test object method that should call your function.

In this statement, you can also instruct UFT to use the function as the default operation for the test object class.

You can register the same function to more than one test object class, using the same operation name for different test object classes, or different names.

After the **RegisterUserFunc** statement runs, your method becomes a recognized method of the specified test object class for the remainder of the run session, or until you unregister the method.

When UFT loads a function library it runs all the statements in the function library. Therefore, if the function you are registering is defined in a function library, it is recommended to include the **RegisterUserFunc** statement in the function library as well so that the method is immediately available for use in any test or component using that function library.

For task details, see "Register the function to a test object class - optional " on page 1049.

Unregistering User-Defined Test Object Methods

Relevant for: GUI tests and components

When you register a method using a **RegisterUserFunc** statement, your method becomes a recognized method of the specified test object class for the remainder of the run session, or until you unregister the method.

If your method overrides a UFT method, unregistering the method resets the method to its normal behavior. Unregistering other methods removes them from the list of methods supported by the test object class.

For task details, see "Unregister the function - optional" on page 1051.

Additional Information for Tests

Unregistering methods is especially important when a reusable action contains registered methods that override UFT methods. For example, if you do not unregister a method that uses a function defined directly within a called action, then the calling test will fail if the registered method is called again in a later action, because it will not be able to find the function definition.

If you register a method within a reusable action, you should unregister the method at the end of the action (and then re-register it at the beginning of the next action if necessary), so that tests calling your action are not affected by the method registration.

If the registered function was defined in a function library, then the calling test may succeed (assuming the function library is associated with the calling test). However, unexpected results may be produced as the author of the calling test may not realize that the called action contained a registered function, and therefore, may use the registered method in later actions, expecting normal UFT behavior.

Unregistering Functions That Were Registered More Than Once

You can re-register the same method to use different user-defined functions without first unregistering the method. However, when you do unregister the method, it resets to its original UFT functionality (or is cleared completely if it was a new method), and not to the previous registration.

Example:

Suppose you enter the following statements:

```
RegisterUserFunc "Link", "Click", "MyClick"
RegisterUserFunc "Link", "Click", "MyClick2"
UnRegisterUserFunc "Link", "Click"
```

After running the **UnRegisterUserFunc** statement, the **Click** method stops using the functionality defined in the MyClick2 function, and returns to the original UFT **Click** functionality, and not to the functionality defined in the MyClick function.

Running an Overriding User-Defined Test Object Method

Relevant for: GUI tests and components

You can register a user-defined function to (temporarily) override the functionality of an existing test object method for a test object class.

When a user-defined function runs instead of the test object method it overrides, if it calls any overridden test object methods, the standard functionality of those methods is used.

When you call the user-defined function directly, if it calls any overridden test object methods, their overriding user-defined functions are used.

Examples

The following scenarios demonstrate various situations that are affected by this functionality:

A Registered User Function That Calls the Test Object Method It Overrides

Suppose you want to report the current value of a Web edit box to the run results before you set a new value for it. You can override the standard UFT Set method with a function that retrieves the current value of an edit box, reports that value to the run results, and then sets the new value of the edit box using the standard Set method.

The function (and its registering line) would look something like this:

```
Function MySet (obj, x)
   dim y
   y = obj.GetROProperty("value")
   Reporter.ReportEvent micDone, "previous value", y
   obj.Set (x)
End Function
RegisterUserFunc "WebEdit", "Set", "MySet"
```

When a test or component step uses the WebEdit.Set method, the overriding MySet function runs, and in turn, calls the original UFT WebEdit Set method.

However, when a test or component step uses the MySet function, the function runs and calls the overridden WebEdit.Set method, running the MySet function once more. This time, MySet calls the original UFT WebEdit Set method.

A Registered User Function That Calls a Test Object Method That Is Overridden by Another Function

Suppose you want to override the VbButton's standard Click method to always perform a double click. In addition, you want to override the standard UFT DblClick method with a function that retrieves the text of the button and reports it to the run results before double-clicking the button.

The functions (and their registering lines) would look something like this:

```
Function MyDblClick (obj, x, y, button)
   dim button_name
   button_name = obj.GetROProperty("text")
   Reporter.ReportEvent micDone, "Clicking", button_name
   obj.DblClick x, y, button
End Function
RegisterUserFunc "VbButton", "DblClick", "MyDblClick"
Function MyClick (obj, x, y, button)
   obj.DblClick x, y, button
End Function
RegisterUserFunc "VbButton", "Click", "MyClick"
```

When a test or component step uses the VbButton.Click method, the overriding MyClick function runs. In this situation, MyClick will then run the original UFT VbButton DblClick method.

When a test or component step uses the MyClick function, the function runs and calls the overridden VbButton.DblClick method, running MyDblClick. MyDblClick reports the button text to the run results and then calls the original UFT VbButton DblClick method.

To ensure that the MyClick function always runs the overridden behavior for DblClick method, you could call MyDblClick directly within MyClick.

Loading Function Libraries During a Run Session

Relevant for: GUI tests and components

If you decide not to associate a function library (any VBScript file) with a test, but do want to be able to call its functions, subroutines, and so forth, you can do so by loading the function library during the run session.

Similarly, if you want to call a function that is not stored in an action in your test or in an associated function library, store it in an independent VBScript file, and load that function library during the run session.

To load a function library during a run session, insert a **LoadFunctionLibrary** statement or **ExecuteFile** statement in your action, scripted component, or function library. When you run the test, this statement runs all global code in the specified function library, making all definitions in the file available for use. For details on these statements, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*, available from the Unified Functional Testing Help.

The following table describes the differences between using each of these statements:

LoadFunctionLibrary	ExecuteFile
In a test: After you run a **LoadFunctionLibrary** statement, the functions in the file are available to your entire test, until the end of the run session. **In a component:** **LoadFunctionLibrary** works in the same way as **ExecuteFile**. After you run the statement, the functions in the file are available only within the scope of the calling component.	After you run an **ExecuteFile** statement, you can call the functions in the loaded file only within the scope of the calling action or component.
LoadFunctionLibrary enables you to debug the functions in the function library during run-time.	You cannot debug a file that is called using an **ExecuteFile** statement, or any of the functions contained in the file. In addition, when debugging a test or component that contains an **ExecuteFile** statement, the execution marker may not be correctly displayed.

If you want functions in a function library (VBScript file) to always be available to your test or component, associate the function library with your test or application area. For details, see "Associated Function Libraries" on page 1031.

Tasks

How to Create and Manage Function Libraries

Relevant for: GUI tests and components

This task describes the different activities you can perform to manage function libraries in UFT. If a function is stored in a function library, you must associate the function library with a test or a component's application area before you can call the function from the test or component.

This task includes the following steps:

- "Create a function library" below
- "Open a function library" on the next page
- "Edit a function library" on the next page
- "Enable editing for a read only function library " on the next page
- "Debug a function in a function library" on the next page
- "Associate a function library with a test or an application area" on page 1043
- "Load a function library dynamically during a run session" on page 1043

Create a function library

Do one of the following:

- Click the **New** button down arrow and select **Function Library**. The "Open/New <Document>/<Resource> Dialog Box" (described on page 156) opens, enabling you to specify the location to save the function library.

  > **Note:** A newly created function library is not automatically associated to the current test or component. Therefore, the new function library is not displayed in the Solution Explorer.

- Create a new function library file from the Application Area (for components only).

  This also associates the function library with the application area, displaying it in the Solution Explorer. For details, see "Function Libraries Pane (Application Area)" on page 2102.

- Create a new function library file from the Test Resources module in ALM. For details, see the *HP Application Lifecycle Management User Guide*.

- Create VBScript function libraries outside of UFT in any editor and save them with a .qfl, .vbs, or .txt extension.

Open a function library

Do one of the following:

- Click the **Open** button down arrow and select **Function Library**. The "Open/New <Document>/<Resource> Dialog Box" (described on page 156) opens, enabling you to open a function library from the file system or from an ALM project.
- If the function library was recently created or opened, select it from the **Recent Files** list in the **File** menu.
- If the function library is associated with the open test, component, or application area, you can also open it as follows:
  - In the Solution Explorer, double-click the function library, or right-click the function library and select **Open Function Library**.
  - In the Toolbox pane, double-click the function library, or right-click the function library and select **Open Resource**.

You can choose to open a function library in edit mode or read-only mode. For details, see "Open/New <Document>/<Resource> Dialog Box" on page 156.

> **Tip:** To open a function library, you must have read or read-write permissions for the file.

Edit a function library

For details, see "How to Edit a Function Library" on the next page.

Enable editing for a read only function library

Right-click the function library tab in the document pane and select **Enable Editing**. You can now edit the function library.

> **Note:**
>
> - You cannot enable editing if the function library is locked by another user or checked in to an ALM project.
> - During a debug session, all documents (such as tests or components and function libraries) are read-only. To edit a document during a debug session, you must first stop the debug session.

Debug a function in a function library

1. Associate the function library with a test or component (for a component, you do this via its application area).
2. In your test or component, insert a call to a function defined in the function library.

3. Place a breakpoint at the beginning of the function in the function library.

4. Run the test or component. When the function is called, UFT pauses the run session at the beginning of the function, enabling you to debug it.

   For details, see "Debugging Tests and Components" on page 674.

   > **Note:**
   >
   > - During a debug session, all documents are read-only and cannot be edited. To edit a document during a debug session, you must first stop the debug session.
   >
   > - You cannot debug a file that is called using an **ExecuteFile** statement, or any of the functions contained in the file. In addition, when debugging a test that contains an **ExecuteFile** statement, the execution marker may not be correctly displayed.
   >
   >   To debug a dynamically loaded function library, use a **LoadFunctionLibrary** statement to load it instead of an **ExecuteFile** statement.

Associate a function library with a test or an application area

For details, see "How to Manage Function Library Associations" on page 1045.

Load a function library dynamically during a run session

Add a the **LoadFunctionLibrary** statement to your action, scripted component, or associated function library. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing* and "Loading Function Libraries During a Run Session" on page 1039.

> **Tip:** To include the same **LoadFunctionLibrary** statement in every action you create, you can add the statement to an action template. For details, see "Create an action template" on page 886.

How to Edit a Function Library

Relevant for: GUI tests and components

This task describes how to edit a function library using the UFT editing features that are available in the Editor.

> **Note:** This task is part of a higher-level task. For details, see "How to Create and Manage Function Libraries" on page 1041.

This task includes the following steps:

- "Add steps manually" below
- "Add steps using the Step Generator (tests and scripted components only)" below
- "Drag and drop a function" below
- "Check the syntax of the code in your function library" below

Add steps manually

When writing the steps in your function library, you can use standard VBScript statements as well as any UFT reserved objects or test objects, and the methods available for these objects.

UFT applies the same editing settings to your function library as is does to content in the Editor of an action. For details, see "Editing Text and Code Documents" on page 305.

Statement completion functionality is available for all functions defined in your action or component and for public functions defined in associated function libraries.

> **Note:** In function libraries, the statement completion feature does not enable you to view test object names or collections because function libraries are not directly associated with object repositories.

Add steps using the Step Generator (tests and scripted components only)

The "Step Generator Dialog Box" (described on page 983) enables you to add steps that contain **reserved objects** (the objects that UFT supplies for enhancement purposes, such as utility objects), VBScript functions (such as **MsgBox**), utility statements (such as **Wait**), and user-defined functions that are defined in the same function library.

Drag and drop a function

You can drag and drop a function (or part of it) within the same document, or from one document to another.

To drag and drop a function from one document to another:

1. Separate the tabbed documents into separate document panes by dragging one document by its tab. You can then optionally dock the document in another location in UFT.
2. Select the relevant lines and drag and drop them from one document to another.

Check the syntax of the code in your function library

Select **Design > Check Syntax**. UFT checks the syntax of all the code in all open function libraries and in function libraries associated with your tests. This may include function definitions and other VBScript statements.

> **Tip:** For details on using VBScript, see "Basic VBScript Syntax" on page 1013.

How to Manage Function Library Associations

Relevant for: GUI tests and components

This task describes the different ways that you can associate a function library with a test or application area or modify existing associations.

> **Note:** This task is part of a higher-level task. For details, see "How to Create and Manage Function Libraries" on page 1041.

This task contains the following steps:

- "View the list of function libraries associated with a test or component" below
- "Associate the currently active function library with a test or an application area" below
- "Associate a function library with a test using the Test Settings dialog box (tests only)" on the next page
- "Associate a function library with a test using the Solution Explorer pane (tests only)" on page 1047
- "Associate a function library with an application area using the Function Libraries pane (components only)" on page 1047
- "Modify the priority of an associated function library" on page 1047
- "Remove a function library association" on page 1048
- "Specify default function libraries for all new tests (tests only)" on page 1048

View the list of function libraries associated with a test or component

Do one of the following:

- In the Solution Explorer pane, expand the **Function Libraries** node within the relevant test or component's node.
- Select the test or component whose associated function libraries you want to view, and then select **File > Settings > Resources**. The Resources pane of the Test/Business Component dialog box opens.

Associate the currently active function library with a test or an application area

1. Make sure that the test or application area with which you want to associate the function library is included in your open solution.

2. Create or open a function library in UFT. For details, see "How to Create and Manage Function

Libraries" on page 1041.

3. Save the function library in the file system (for tests only) or in your ALM project.

4. In UFT, do one of the following:

   - right-click the function library document tab and select **Associate Library '<Function Library>' with '<Test/Application Area>'**.

   - right-click the test or application area name node in the Solution Explorer and select **Associate Function Library**.

Associate a function library with a test using the Test Settings dialog box (tests only)

1. Create or open a test.

   Note: You can access or create a test using a relative path. If you enter a relative path, UFT searches for the test in the folders listed in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** pane). For details, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544 and "Relative Paths in UFT for GUI Testing" on page 130.

2. In the Test Settings dialog box (**File > Settings**), click the **Resources** node.

3. In the **Associated function libraries** list, click the **Add** button +, UFT displays a browse button enabling you to browse to a function library in the file system. If you are connected to an ALM project, UFT also adds [ALM] to the file path, indicating that you can browse to a function library either in your ALM project or in the file system.

   Tip: If you want to add a file from your ALM project but are not connected to ALM, press and hold the SHIFT key and click the **Add** button +. UFT adds [ALM], and you can enter the path manually. If you do, make sure there is a space after [ALM]. For example: [ALM] Subject\Tests

   Note that UFT searches ALM project folders only when you are connected to the corresponding ALM project.

4. Select the function library you want to associate with your test and click **Open**.

Associate a function library with a test using the Solution Explorer pane (tests only)

In the Solution Explorer pane, do one of the following:

- Right click a GUI**<test name>** node and select **Associate Function Library.**
- Right-click the **Function Libraries** node within the relevant test's node in the tree and select **Associate Function Library**.

The "Open/New <Document>/<Resource> Dialog Box" (described on page 156) opens.

The function library that you select is associated with the test and displayed as a node under the **Function Libraries** node in the tree.

> **Note:** A test node contains the **Function Libraries** node only if at least one function library is associated with the test. If the relevant test does not contain a **Function Libraries** child node, use one of the other association methods described in this task to associate the first function library.

Associate a function library with an application area using the Function Libraries pane (components only)

1. In UFT, open your application area and click the **Function Libraries** button on the sidebar.
2. In the **Associated function libraries** list, click the **Add** button +. UFT displays a browse button enabling you to browse to a function library in your ALM project.
3. Select the function library you want to associate with your application area and click **Open**.

Modify the priority of an associated function library

For tests, do one of the following:

- In the Solution Explorer, expand the Function Libraries node for your test or application area, right-click the function library you want to prioritize and select **Move up** or **Move down**. For details, see "Solution Explorer Pane User Interface" on page 478.
- In the list of associated function libraries in the Resources pane of the Test Settings dialog box (for tests) or the Function Libraries pane (for application areas), select the function library you want to prioritize and use the **Up** and **Down** arrows ↑ ↓.

  For details, see:

  - **For tests:** "Resources Pane (Test/Business Component Settings Dialog Box) " on page 603
  - **For components:** "Function Libraries Pane (Application Area)" on page 2102.

Remove a function library association

Do one of the following:

- In the Solution Explorer, expand the Function Libraries node for your test or application area, right-click the function library and select **Remove Function Library from List**, or select the function library and press the DELETE key. For details, see "Solution Explorer Pane User Interface" on page 478.

- In the list of associated function libraries in the Resources pane of the Test Settings dialog box (for tests) or the Function Libraries pane (for application areas), select the function library you want to remove and click the **Remove** button ☒.

  For details, see:

  - **For tests:** "Resources Pane (Test/Business Component Settings Dialog Box) " on page 603.
  - **For components:** "Function Libraries Pane (Application Area)" on page 2102.

- Open an associated function library in UFT. For details, see "How to Create and Manage Function Libraries" on page 1041.

  Right-click the function library document tab and select **Dissociate Library '<Function Library>' from '<Test/Application Area>'**.

Specify default function libraries for all new tests (tests only)

In the "Resources Pane (Test/Business Component Settings Dialog Box) ", described on page 603, create the list of associated function libraries that you want to use for every newly created test, and click the **Set as Default** button. (This does not affect existing tests.)

How to Create and Work with a User-Defined Function

Relevant for: GUI tests and components

This task describes how to create and work with a user-defined function.

This task includes the following steps:

- "Prerequisites - Open the function library or test " on the next page
- "Create the function" on the next page
- "Register the function to a test object class - optional " on the next page
- "Associate the function library with a test or application area" on page 1050
- "Call the function" on page 1050
- "Navigate to the function's definition - optional" on page 1051
- "Unregister the function - optional" on page 1051

1. **Prerequisites - Open the function library or test**

   a. **For tests:** Determine whether you want to store the function in an action or in a function library.

      - If you insert the function in a function library, the function will be accessible to any associated test.

      - If you insert the function directly in an action in the Editor, it can be called only from within the specific action.

   b. Create a new function library or action, open an existing one, or click on the tab of an open function library or action to bring it into focus.

2. **Create the function**

   You can define functions manually or using the Function Definition Generator, which creates the basic function definition for you automatically.

   Even if you prefer to define functions manually, you may still want to use the Function Definition Generator to view the syntax required to add header information, register a function to a test object class, or set the function as the default method for that test object class. For an in-depth view of the required syntax, you can define a function using the Function Definition Generator and experiment with the various options.

   For details, see "Function Definition Generator Dialog Box" on page 1057.

   When writing the code for your function, consider the issues listed in "Troubleshooting and Limitations - Function Libraries" on page 1065. For details on using VBScript, see the Microsoft VBScript Language Reference (**Help > HP Unified Functional Testing Help > VBScript Reference > VBScript**).

   > **Note:**
   >
   > - If you want to register the function to a test object class, define it as a public function, and make sure that it expects the test object as the first argument.
   >
   > - If you want to override an existing test object method, make sure that after the test object argument, your function accepts the same number of arguments as the method it overrides.

3. **Register the function to a test object class - optional**

   You can register your function as a new method for the test object class, or you can register it using an existing method name to (temporarily) override the existing functionality of the specified method.

   You can perform this step manually, or using the "Function Definition Generator Dialog Box" (described on page 1057):

- Add a **RegisterUserFunc** statement in your action or function library. The name of the test object operation you register cannot contain spaces. In this statement, you can also instruct UFT to use the function as the default operation for the test object class.

  > **Example:**
  >
  > RegisterUserFunc "WebEdit", "MySet", "MySetFunc", True
  >
  > After this statement runs (during the run session), the MySet method (operation) is added to the WebEdit test object class using the MySetFunc user-defined function, and defined to be the default operation (as specified in the last argument of the statement).
  >
  > If you or the Subject Matter Expert choose the WebEdit test object from the **Item** list in the Keyword View, the MySet operation is selected automatically in the **Operation** column. It is also displayed in the **Operation** list together with other registered and out of the box operations for the WebEdit test object.

  For syntax and other examples, see the **Utility Objects > Utility Statements** section of the *HP UFT Object Model Reference for GUI Testing*, available from the Unified Functional Testing Help.

- If you use the "Function Definition Generator Dialog Box" (described on page 1057) to create your function definition, a RegisterUserFunc statement is automatically added immediately after the definition if you select the **Register to a test object** option.

  You can add additional RegisterUserFunc statements manually to register the function to additional test object classes.

  > **Tip:** If the function you are registering is defined in a function library, it is recommended to include the RegisterUserFunc statement in the function library as well so that the method will be immediately available for use in any test or component using that function library.

4. Associate the function library with a test or application area

If you inserted the code in a function library, you must associate the function library with a test or application area to enable tests and components to access to the user-defined functions. For details, see "How to Manage Function Library Associations" on page 1045.

Alternatively, you can add a **LoadFunctionLibrary** statement to your test or component to load the function library during the run session and access its functions. For details, see the **Utility Objects > Utility Statements** section of the *HP UFT Object Model Reference for GUI Testing*.

5. Call the function

In your test, component, or function library, do one or both of the following:

- Create steps that call your user-defined function as a global function.
- Run the user-defined function by calling the test object method to which it is registered.

6. **Navigate to the function's definition - optional**

   You can navigate directly from a function call to the function's definition.

   a. In the Editor, in an action, click in the step containing the relevant function.

   b. Perform one of the following:

      - Select **Search > Go to > Definition**.
      - Right-click the step and select **Go to Definition** from the context menu.

        UFT activates the relevant document (if the function definition is located in a function library) and positions the cursor at the beginning of the function definition.

7. **Unregister the function - optional**

   If you do not want your function to remain registered until the end of the run session, add an **UnRegisterUserFunc** statement in your test or function library. For syntax and an example, see the **Utility Objects > Utility Statements** section of the *HP UFT Object Model Reference for GUI Testing*.

   For tests: If you register a method within a reusable action, you should unregister the method at the end of the action (and then re-register it at the beginning of the next action if necessary), so that tests calling your action are not affected by the method registration.

How to Create and Register a User-Defined Function Using the Function Definition Generator

Relevant for: GUI tests and components

This task provides a general description of each step required to create a user-defined function and register it to a test object class using the "Function Definition Generator Dialog Box" (described on page 1057). This task also includes examples for some of the steps.

> **Note:** This task is part of a higher-level task. For details, see "How to Create and Work with a User-Defined Function" on page 1048.

This task includes the following steps:

- "Prerequisite - Open the function library or action" on the next page
- "Open the Function Definition Generator" on the next page

- "Specify the details for the function definition" below
- "Register the function to a test object class - optional" below
- "Add arguments to the function - optional" on the next page
- "Add documentation details to the function - optional" on the next page
- "Preview the function definition before finalizing it" on page 1054
- "Insert the function in your active document " on page 1055
- "Add the content (code) of the function" on page 1055
- "Results" on page 1055

1. **Prerequisite - Open the function library or action**

   Make sure that the function library or action in which you want to insert the function definition is the active document. (You can click the function library or action document's tab to bring it into focus.) This is because the Function Definition Generator inserts the function in the currently active document after you finish defining it.

2. **Open the Function Definition Generator**

   Select **Design > Function Definition Generator**. The "Function Definition Generator Dialog Box" (described on page 1057) opens.

3. **Specify the details for the function definition**

   Specify the function's name, type, and scope in the "Function Definition Area" on page 1059.

   > **Example:**
   >
   > If you want to define a function that verifies the value of a specified property, you might name it VerifyProperty and define it as a public function so that it can be called from any associated test or component. (If you define it as private, the function can be called only from elsewhere in the same function library. Private functions cannot be registered to a test object class.)

4. **Register the function to a test object class - optional**

   If you defined a public function and you want to register the function as a test object operation, do the following in the "Registration Area" (described on page 1061):

a. Select the **Register to a test object** check box.

b. Select the test object from the list of available objects.

> **Example:**
>
> For the sample **VerifyProperty** function, you might want to register it to the Link test object.

c. Enter the name of a new operation that you want to add to the test object class, or select an existing operation to specify the operation that you want to override its standard functionality. The name of the method cannot contain spaces.

> **Example:**
>
> For the sample **VerifyProperty** function, you might want to define a new **VerifyProperty** operation.

d. Optionally, specify that this operation is the default operation for test objects of this type.

> **Tip:** If you choose not to register your function at this time, you can manually register it later by adding a **RegisterUserFunc** statement as described in "Register the function to a test object class - optional " on page 1049. You can also add additional **RegisterUserFunc** statements, to register the function to additional test object classes.

5. Add arguments to the function - optional

Specify any arguments that are required for your function to run correctly. For details, see "Arguments Area" on page 1062.

> **Example:**
>
> For the **VerifyProperty** function, registered to a test object class in the example for the previous step, you may want to assign the arguments prop_name (the name of the property to check) and expected_value (the expected value of the property), in addition to the first argument, **test_object**.

6. Add documentation details to the function - optional

Define **Description** and **Documentation** strings for the test object operation. For details, see "Additional Information Area" on page 1063.

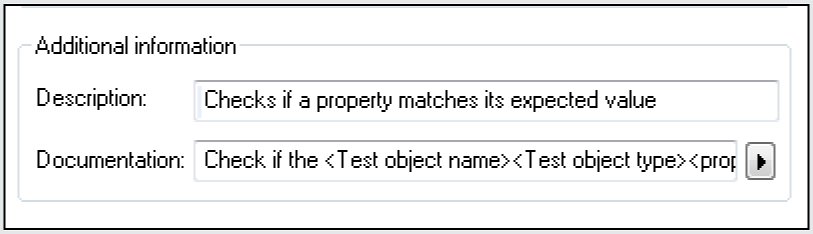
Example:

- For the sample **VerifyProperty** function, you may want to provide the following description:

  Checks whether a property value matches the actual value.

- If you were checking a link to "HP" from a search engine, you might define the following documentation using the Function Definition Generator:

  `@Documentation Check if the <Test object name> <Test object type> <prop_name> value matches the expected value: <expected_value>.

  Additional information

  Description: Checks if a property matches its expected value

  Documentation: Check if the <Test object name><Test object type><prop

  In the Keyword View, after you create a step that calls the **VerifyProperty** operation and choose values for the arguments, the above documentation might be displayed as follows:

  Check if the "Management Software" link "text" value matches the expected value: "Business Technology Optimization (BTO) Software".

7. Preview the function definition before finalizing it

As you add information to the Function Definition Generator, the **Preview** area displays the emerging function definition. You can review your function and make any changes, as needed, in the various areas of the dialog box. For details, see "Bottom Area" on page 1064.

Example:

After defining the **VerifyProperty** function as described in the previous steps, the **Preview** area displays the following code:

```
‘@Description Checks whether a property matches its expected value
‘@Documentation Check whether the <Test object name> <Test object type> <prop_name> value matches the expected value: <expected_value>.
Public Function VerifyProperty (test_object, prop_name, expected_value)
    ‘TODO: add function body here
```

```
End Function
RegisterUserFunc "Link", "VerifyProperty", "VerifyProperty"
```

8. **Insert the function in your active document**

   a. To generate an additional function definition after inserting this one, select **Insert another function definition**. The **OK** button changes to **Insert**.

   b. Click **OK** or **Insert**. UFT inserts the generated VBScript code in the active document.

9. **Add the content (code) of the function**

   To finalize the function, add content to the function code, as required, replacing the TODO comment.

   When writing the code for your function, consider the issues listed in "Troubleshooting and Limitations - Function Libraries" on page 1065. For details on using VBScript, see the Microsoft VBScript Language Reference (**Help > HP Unified Functional Testing Help > VBScript Reference > VBScript**).

   Example:

   For example, if you want the function to verify whether the expected value of a property matches the actual property value of a specific test object, you might add the following to the body of the function:

```
Dim actual_value
  ' Get the actual property value
  actual_value = obj.GetROProperty(prop_name)
  ' Compare the actual value to the expected value
  If actual_value = expected_value Then
    Reporter.ReportEvent micPass, "VerifyProperty Succeeded", "The " & prop_name & " ex
pected value: " & expected_value & " matches the actual value"
    VerifyProperty = True
  Else
    Reporter.ReportEvent micFail, "VerifyProperty Failed", "The " & prop_name & " expected
value: " & expected_value & " does not match the actual value: " & actual_value
    VerifyProperty = False
  End If
```

10. **Results**

   - If you inserted the function in a function library, the function is now accessible to any associated test or component.

     If you inserted the function directly in an action in the Editor, it can be called only from within the specific action.

To associate the function library with additional tests or application areas, see "How to Manage Function Library Associations" on page 1045.

- If you registered the function to a test object, the function is displayed in the list of available operations for the test object, as well as in the general **Operation** list in the Step Generator (for function libraries). For details on where you can view the list of operations available for a test object, see "Registered User-Defined Functions" on page 1035.

  You can add additional **RegisterUserFunc** statements, to register the function to additional test object classes. For details, see "Register the function to a test object class - optional " on page 1049.

- If you specified that the function is the default operation for the test object, the function is used as the test object operation, by default, when you create steps with that test object. For details, see "Registered User-Defined Functions" on page 1035.

- If you defined a description and a **Documentation** string for the test object operation, they are displayed in the Keyword View and the Step Generator.

Reference

Function Definition Generator Dialog Box

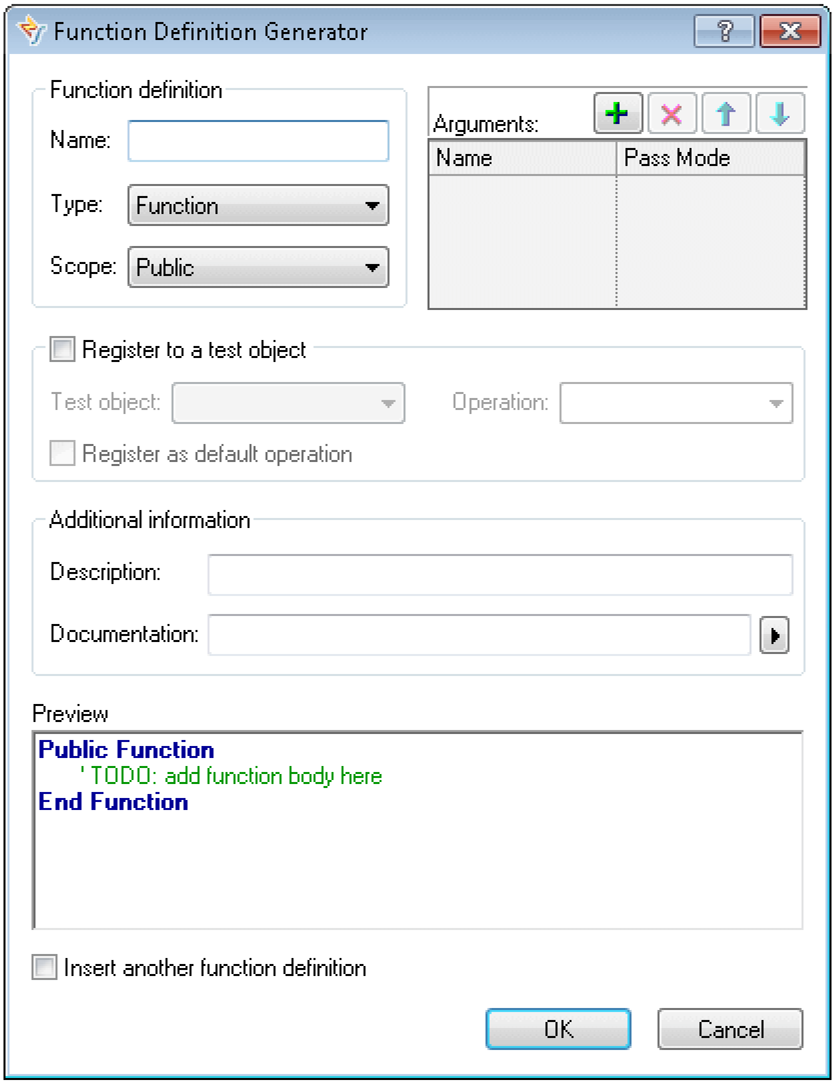
Relevant for: GUI tests and components

This dialog box enables you to:

- Generate definitions for new user-defined functions.
- Add header information to the function you create.
- Register the functions to a test object class, if needed.

You fill in the required information and the Function Definition Generator creates the basic function definition for you. You complete the function by adding its content (code).

For a task that describes the order of working with this dialog box and provides examples, see "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1051.



Type: Function

Scope: Public

```
Public Function
    ' TODO: add function body here
End Function
```

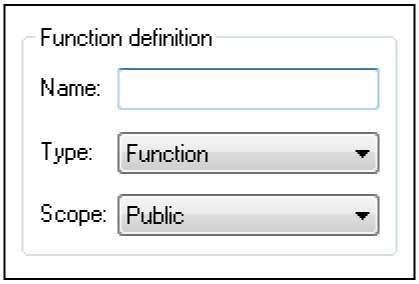
To access	Select **Design > Function Definition Generator**
Important information	Before you open the dialog box, make sure that the function library or action (displayed in the Editor) in which you want to insert the function definition is the active document. (You can click the function library or action document's tab to bring it into focus.) This is because the Function Definition Generator inserts the function in the currently active document when you click **OK**.
Relevant tasks	• "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1051 • "How to Create and Work with a User-Defined Function" on page 1048

See also	- "User-Defined Functions" on page 1032 - "User-Defined Function Storage and Access" on page 1034 - "Registered User-Defined Functions" on page 1035

This dialog box contains the following key areas:

- "Function Definition Area" below
- "Registration Area" on page 1061
- "Arguments Area" on page 1062
- "Additional Information Area" on page 1063
- "Bottom Area" on page 1064

Function Definition Area


Function definition

Name:

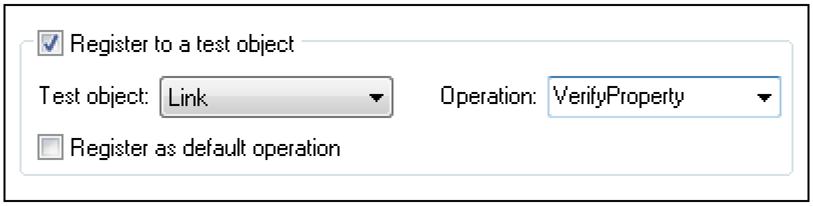
Type: Function

Scope: Public

User interface elements are described below:

UI Elements	Description
Name	A name for the new function. The name should clearly indicate what the operation does so that it can be easily selected from the Step Generator or in the Keyword View. For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269. **Note:** Do not use any of the built-in function names (for example, **GetLastError**, **MsgBox**, or **Print**). For a list of built-in functions, see the **Built-in functions** list in the Step Generator (**Design > Step Generator**). Try to give each function a name that is unique within the resources associated with a specific test or application area. For details, see "User-Defined Function Names" on page 1033.
Type	The type of function. **Possible values:** • **Function** • **Sub** (subroutine)
Scope	The scope of the function. **Possible values:** • **Public.** The function can be called by any test associated with this function library and any component whose application area is associated with this function library. • **Private.** The function can be called only from elsewhere in the same function library. **Default value:** Public **Note:** • Only public functions can be registered to a test object class. • If you create a user-defined function manually and do not define the scope as **Public** or **Private**, it is treated as a public function, by default.

Registration Area


☑ Register to a test object

Test object: Link

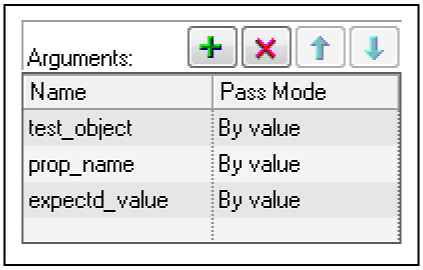
Operation: VerifyProperty

□ Register as default operation

User interface elements are described below:

UI Elements	Description
Register to a test object	Indicates whether to register this function as an operation for a UFT test object class. All user-defined functions are available as global operations. If you register the function to a test object class, it can be also be called by test objects of that class, and is displayed in the list of available operations for such test objects. For details, see "Registered User-Defined Functions" on page 1035. **Note:** If you select this option, then when the Function Definition Generator creates your function definition, it also automatically adds a **RegisterUserFunc** statement with the correct argument values immediately after the definition.
Test object	The test object class (type) to which you want to register the function. **Available:** When **Register to a test object** is selected
Operation	The operation name to use for this function. You can select an existing test object operation to override its functionality, or enter a name for a new operation to add to the test object class. Do not include spaces in the test object operation name. **Available:** When **Register to a test object** is selected
Register as default operation	Indicates whether the function should be the default operation for the test object class. This instructs UFT to use the function as the test object operation, by default, when you create steps for test objects of the specified type. For details, see "Registered User-Defined Functions" on page 1035. **Note:** If you select this option, then when the Function Definition Generator creates your function definition, it specifies the value **True** as the fourth argument of the **RegisterUserFunc** statement. **Available:** When **Register to a test object** is selected

Arguments Area



Arguments:

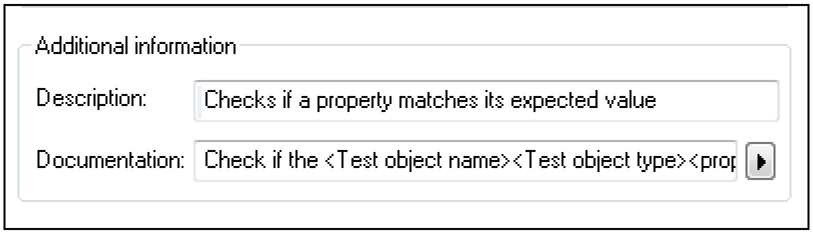
Name	Pass Mode
test_object	By value
prop_name	By value
expectd_value	By value

Important information	When calling a function that is registered to a test object class, UFT passes the test object as the first argument. Therefore: - If you select the **Register to a test object** check box, the Function Definition Generator automatically adds the argument, **test_object**, as the first argument in this area. - If you clear the **Register to a test object** check box, the default **test_object** argument is automatically removed from this area (unless you renamed it).

User interface elements are described below:

UI Elements	Description
+ ✕ ↑ ↓	A toolbar that enables you to add, remove, or re-arrange arguments in the list. You can add as many arguments as you want to the list. **Caution:** - If you are registering the function to a test object class, do not remove the **test_object** argument, change its location in the list, or modify its **Pass Mode**. - If you are registering the function to override an existing test object method, then after the test object argument, the function must have the same number of arguments as the method it overrides.
Name	The argument name. The name should clearly indicate the value that needs to be entered for the argument. For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269.
Pass Mode	Specifies whether the argument is passed to the function **By value** or **By reference** during run-time.

Additional Information Area



Additional information

Description: Checks if a property matches its expected value

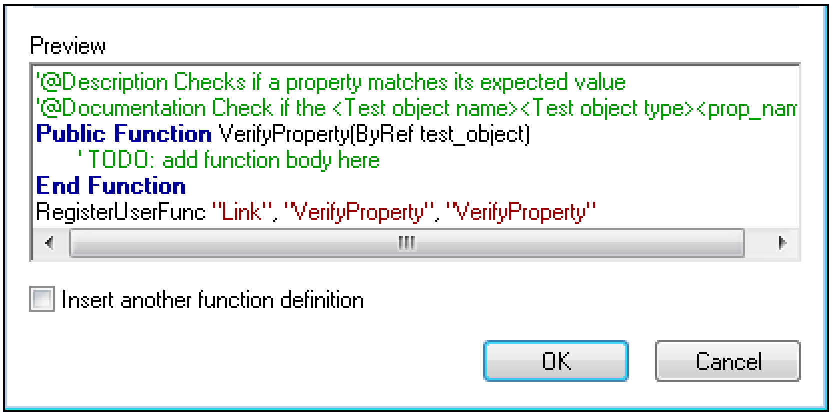
Documentation: Check if the <Test object name><Test object type><prop

The Function Definition Generator can add header information to your user-defined function definitions, based on the information you add in this area.

User interface elements are described below:

UI Elements	Description
Description	The function's description. If you register the function as a test object operation, the description is displayed as a tooltip in UFT when the cursor is positioned over the operation in the Step Generator, in the Keyword View, and when using the statement completion feature. Keep the description text brief and clear.
Documentation	A sentence (in the imperative form) that specifies exactly what a step using your function does. If you register the function as a test object operation, the text that you add here is displayed in the **Step documentation** box of the Step Generator (tests only) and in the **Documentation** column in the Keyword View. Therefore, the sentence must be clear and understandable.
▶	**Insert Documentation Element.** Displays a list that contains the function's arguments, and the items **test object name** and **test object type**, enabling you to include these items in the **Documentation** text. If you use the argument or test object items in the **Documentation** text, they are replaced dynamically by the relevant test object names and types or argument values when you create a step that uses the operation. **Note:** The **Test object name** and **Test object type** items are available in the list only if you selected the **Register to a test object** check box.

Bottom Area



Preview

```
'@Description Checks if a property matches its expected value
'@Documentation Check if the <Test object name><Test object type><prop_nam
Public Function VerifyProperty(ByRef test_object)
	' TODO: add function body here
End Function
RegisterUserFunc "Link", "VerifyProperty", "VerifyProperty"
```

□ Insert another function definition

User interface elements are described below:

UI Elements	Description
Preview	Displays the VBScript code that the Function Definition Generator will add to your active document, based on the information that you enter in the dialog box. The code is displayed in read-only format and includes: • An empty function definition • Header information for the function documentation (if defined) • A **RegisterUserFunc** statement (if you selected **Register to a test object**) The function definition is displayed dynamically, as you enter the information. You can review your function and make any changes, as needed, in the various areas of the dialog box.
Insert another function definition	Specifies whether the dialog box should remain open after you click **OK**, enabling you to define an additional function. If you select this option, the **OK** button changes to **Insert**.
OK / Insert	Inserts the generated VBScript code in the active document and clears the data from the dialog box.

Troubleshooting and Limitations - Function Libraries

Relevant for: GUI tests and components

This section describes troubleshooting and limitations for user-defined functions and function libraries.

- If you define a VBScript class, it can be called only within the UFT action or function library in which you defined it.

  Workaround:

  a. You can use an **ExecuteFile** statement to call a VBScript class defined in an external function library. However, you cannot debug a file that is called using an ExecuteFile statement, or any of the functions contained in the file. In addition, when debugging a test or component that contains an ExecuteFile statement, the execution marker may not be correctly displayed.

  b. In the function library in which you define the class, create a function that sets an object as your class, and returns the object. Then you can call this function from anywhere in your test or component, to assign an object of your class to a variable you define.

  > **Example:**
  >
  > Suppose you want to use the class myClass. In the same function library as you define myClass, define the following function:
  >
  > ```
  > public function myClassGenerator()
  > set myClassGenerator = new myClass
  > end function
  > ```
  >
  > Now, you can use the myClass class by calling myClassGenerator, like this:
  >
  > ```
  > set myClassObject = myClassGenerator()
  > ```
  >
  > myClassGenerator() creates a new myClass object and assigns it to myClassObject.

- You can use the **RegisterUserFunc** statement to register a user-defined function that overrides an existing test object method. You can also register a user-defined function to override a test object method that was created using a UFT Extensibility SDK. If you override this type of test object method, the user-defined function must not (recursively) call the test object method that it overrides.

- By default, steps that use user-defined functions are not included in the run results after a run session. If you want the function to appear in the run results, you must add a **ReportEvent Method** statement to the function code. For example, you may want to provide additional information or to modify the test, component, or business process test status. For details on the

Reporter.ReportEvent statement, see the Utility Objects section of the *HP UFT Object Model Reference for GUI Testing*.

If a step within your user-defined function calls a standard UFT test object method, this step will appear in the run results after the run session. However, you can still add a **Reporter.ReportEvent** statement to the function code to provide additional information and to modify the test, component, or business process test status, if required.

- **For tests:** If you use a partial run or debug option, such as **Run from step** or **Debug from step**, to begin running a test from a point after method registration was performed in a test step (and not in a function library), UFT does not recognize the method registration because it occurred prior to the beginning of the current run session.

- If you delete a function in use from an associated function library, the step using the function will display the ? icon. In subsequent run sessions for the test, component, or business process test, an error will occur when the step using the non-existent function is reached.

- To use an **Option Explicit** statement in a function library associated with your test or component, you must include it in all of the function libraries associated with the test or component. If you include an **Option Explicit** statement in only some of the associated function libraries, UFT ignores all of the **Option Explicit** statements in all function libraries.

  In test actions, you can use **Option Explicit** statements directly without any restrictions.

- Each function library must have unique variables in its global scope. If you have two associated function libraries that define the same variable in the global scope using a Dim statement or define two constants with the same name, the second definition causes a syntax error. If you need to use more than one variable with the same name in the global scope, include a Dim statement only in the last function library (since function libraries are loaded in the reverse order).

- Function libraries that run in the same run session must not contain different definitions for the same class. Make sure that each class is defined in only one location.

- If another user modifies a function library that is referenced by a test or component, or if you modify the function library using an external editor (not UFT), the changes take effect only after the test or component is reopened.

- Function libraries that contain the Cyrillic ' я ' character and are saved in ANSI encoding may be interpreted incorrectly in UFT. For example, the ' я ' character may be interpreted as a newline character, causing the run to fail.

  Workaround: If this problem occurs, use a text editor to convert the function library by saving it in Unicode encoding.

Chapter 38: Integration with API Tests

Relevant for: GUI tests only

This chapter includes:

Concepts

Integrating API Tests and Actions into GUI Tests

Relevant for: GUI tests only

UFT enables you to test your GUI-less applications (also known as headless applications) while testing your GUI. For example, you can test standard Web Services, non-SOAP Web Services, such as REST, and so on.

You can include calls from your GUI test to API tests and actions if UFT is using an HP Unified Functional Testing (also known as **UnifiedFunctionalTesting**) license.

When you insert a call to an API test or action, the call is displayed under the relevant GUI action in the canvas and in the action (displayed in the Editor).

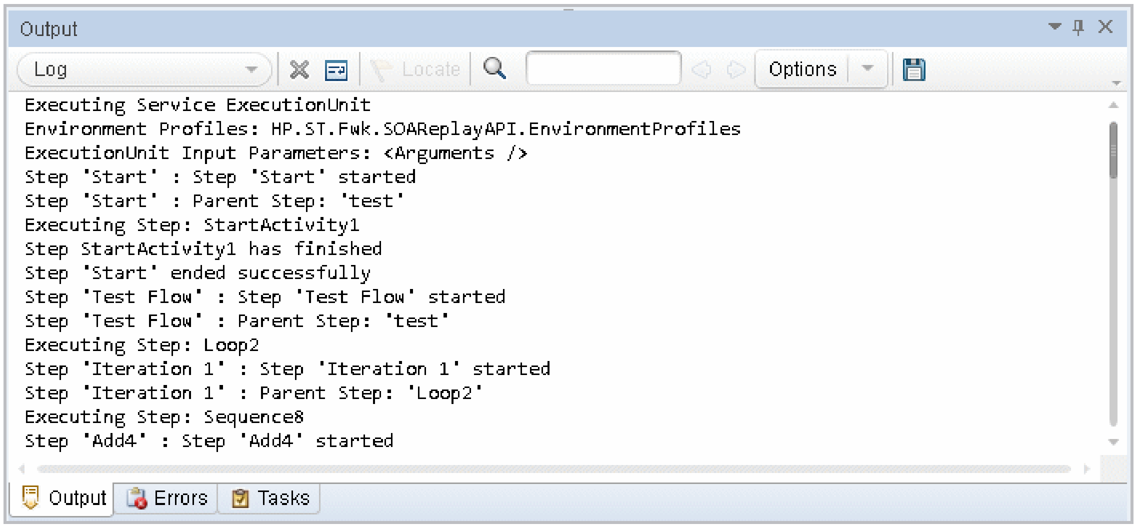
If UFT is connected to an ALM project that contains API tests, you can call an API test or action that is stored in that project. When you run the test, make sure that the UFT client on which you run the test has access to an HP Unified Functional Testing license.

You insert and modify calls to API tests using the "Call to API Test/Action Dialog Box" (described on page 1075). When you insert a call, you can also specify the parameter values that UFT will use when running the test or action. If a parameter has a default value, you can retain it or modify it as needed.

For a use case scenario describing this process, see "Using API Tests in a GUI Test - Use-Case Scenario" on the next page.

How UFT Runs the Called API Test or Action

When a step containing a call to an API test or action is reached, UFT opens the called test and runs it. During the run session, the Output pane displays a real-time log of the steps that are being performed in the API test, as shown in the following example. For details on what is contained in these steps, see "Output Pane Overview" on page 380.



```
Executing Service ExecutionUnit
Environment Profiles: HP.ST.Fwk.SOAReplayAPI.EnvironmentProfiles
ExecutionUnit Input Parameters: <Arguments />
Step 'Start' : Step 'Start' started
Step 'Start' : Parent Step: 'test'
Executing Step: StartActivity1
Step StartActivity1 has finished
Step 'Start' ended successfully
Step 'Test Flow' : Step 'Test Flow' started
Step 'Test Flow' : Parent Step: 'test'
Executing Step: Loop2
Step 'Iteration 1' : Step 'Iteration 1' started
Step 'Iteration 1' : Parent Step: 'Loop2'
Executing Step: Sequence8
Step 'Add4' : Step 'Add4' started
```

Note: UFT must be using an HP Unified Functional Testing (instead of a QuickTest-only) license to run a call to an API test within a GUI test.

If a Unified Functional Testing license type is not available, the run session behavior is different depending on whether the test runs from UFT or from ALM.

- **When running from UFT:** The GUI test runs until the step calling the API test is reached, and then fails.
- **When running from ALM:** If the GUI test contains a call to an API test, UFT does not open and the test does not run.

After the run session, the results display information about the UFT test including the called API test or action. You can view these results in the Run Results Viewer. For details, see the section describing GUI Tests Containing Calls to API Tests (described in the *HP Run Results Viewer User Guide*).

Using API Tests in a GUI Test - Use-Case Scenario

Relevant for: GUI tests only

Suppose you want to test that the API (non GUI, or service) layer of your application performs correctly with the GUI layer of your application. You want to see that when you perform specific steps in the user interface the API (service) processes that accompany the commands given by the user interface are completed correctly.

This use-case scenario describes an example of how to incorporate tests for the API (service) layer of your application into a GUI test. For the purposes of this scenario, you will be using a flight booking application similar to the Mercury Tours Flight GUI and Flight API applications provided with the UFT installation and used in the Tutorial for GUI Testing and Tutorial for API Testing.

> **Note:** For a task related to this scenario, see "How to Integrate API Tests and Actions in GUI Tests" on page 1073.

> **Tip:** For additional information and user interface details, see "Integrating API Tests and Actions into GUI Tests" on page 1068 and "Call to API Test/Action Dialog Box" on page 1075.

In your application, you have four different pages, which correspond to the different tasks involved in booking a flight:

- Logging in to the booking site (**Login Page**)
- Finding flight options based on customer selections for departure city, arrival city, and flight date (**Flight Finder Page**)
- Selecting a flight from the list of flight options (**Select Flight Page**)
- Booking and confirming a customer's flight selection. (**Book Flight Page**)

In addition, your application has a number of API processes to help the application process flight booking requests:

- Finding user login credentials in the database (**Login** operation)
- Searching for a list of all available flights and displaying the flight list (**FindFlights** operation)
- Creating a flight order (**CreateFlight** operation)
- Confirming a flight booking (**ConfirmBookFlight** operation)
- Updating a flight order (**UpdateFlightOrder** operation)
- Deleting a flight order (**DeleteFlightOrder** operation)
- Deleting all flight orders (**DeleteAllFlightOrders** operation)

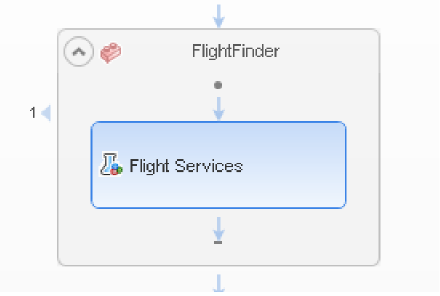
You create a separate GUI action for each application page, giving it the same name as the page name. You also create a separate test for each API process, naming each test with the process name.

> **Note:** For details on creating API tests, see "How to Create an API Test" on page 1598

In order to fully test your application, you decide to place an API test after each GUI test.The API test checks to see whether the API processes run by that specific application's page (**Login**, **Flight Finder**, **Select Flight**, or **Book Flight**) are working correctly.

In your GUI actions, you insert a call to the corresponding API test using **Design > Call to Existing API Test/Action** or by clicking the **Insert Call to Action** down arrow and selecting **Call to Existing API Test/Action**.

The API test appears is displayed as nested inside the GUI action:

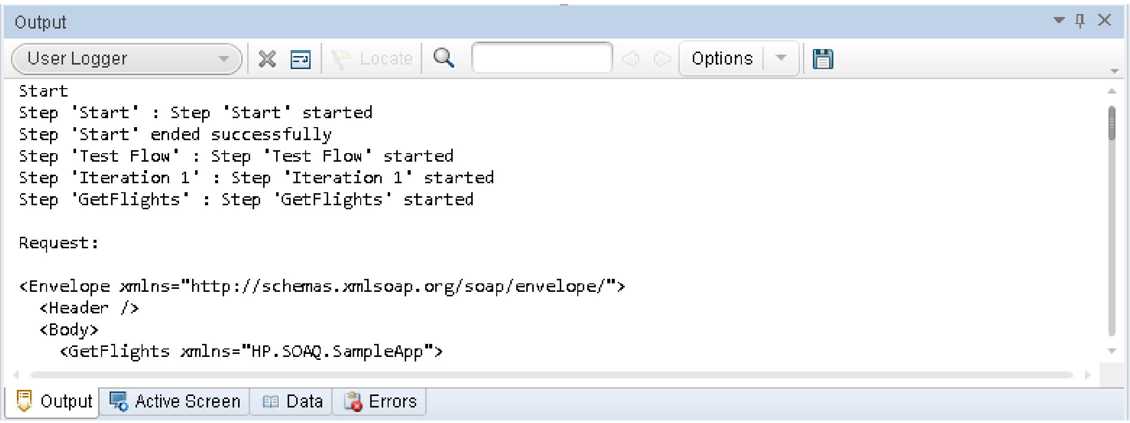


After you insert all the calls to the corresponding API tests, you have a call to an API test inside each of your GUI test actions as listed in the table below:

GUI Test Action Name	Calls API test:
Login Page	Login
Flight Finder Page	FindFlights
Select Flight Page	CreateFlight
Book Flight Page	ConfirmBookFlig

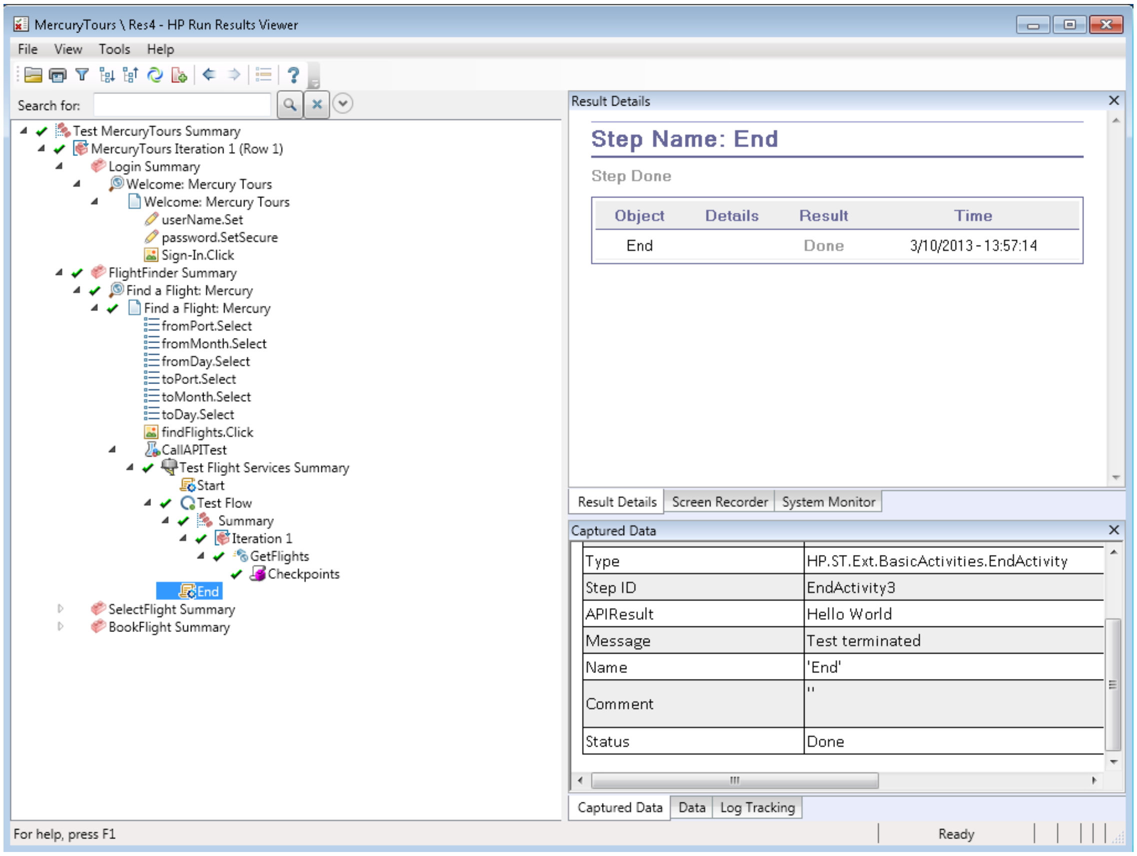
Note: If you want to pass data from a API test to use in an GUI test, You must create a test output parameter in your API test. For details, see "Add Input/Output Property/Parameter Dialog Box (API Testing)" on page 1775

After adding the necessary calls to your API tests, you can run the test. The GUI test executes each step the user interface in the flight booking application user interface. For each API test call in the GUI test, UFT compiles the API test and runs it. UFT displays the API test steps running in the Output pane.



```
Start
Step 'Start' : Step 'Start' started
Step 'Start' ended successfully
Step 'Test Flow' : Step 'Test Flow' started
Step 'Iteration 1' : Step 'Iteration 1' started
Step 'GetFlights' : Step 'GetFlights' started

Request:

<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Header />
  <Body>
    <GetFlights xmlns="HP.SOAQ.SampleApp">
```

After the test run is complete, you can check the test results for the GUI test, including calls to each of the API tests. The Run Results Viewer shows the completion and pass/fail status for each step.



```
Test MercuryTours Summary
  MercuryTours Iteration 1 (Row 1)
    Login Summary
      Welcome: Mercury Tours
        Welcome: Mercury Tours
          userName.Set
          password.SetSecure
          Sign-In.Click
    FlightFinder Summary
      Find a Flight: Mercury
        Find a Flight: Mercury
          fromPort.Select
          fromMonth.Select
          fromDay.Select
          toPort.Select
          toMonth.Select
          toDay.Select
          findFlights.Click
          CallAPITest
            Test Flight Services Summary
              Start
              Test Flow
                Summary
                  Iteration 1
                    GetFlights
                      Checkpoints
              End
    SelectFlight Summary
    BookFlight Summary
```

Step Name: End

Step Done

Object	Details	Result	Time
End		Done	3/10/2013 - 13:57:14

Type	HP.ST.Ext.BasicActivities.EndActivity
Step ID	EndActivity3
APIResult	Hello World
Message	Test terminated
Name	'End'
Comment	''
Status	Done

Using this, you can create composite tests of your application, enabling you to simultaneously test the GUI and API (service) sides of your application by running a single test.

Tasks

How to Integrate API Tests and Actions in GUI Tests

Relevant for: GUI tests only

This task describes how to insert a call to an API test or action.

This task includes the following steps:

- "Prerequisites" below
- "Insert or modify a call to an API test" below
- "Use value of the API test parameters in a GUI test" on the next page

1. Prerequisites

A **UnifiedFunctionalTesting** license must be loaded. For details, see "Add-in Manager Dialog Box" on page 118.

2. Insert or modify a call to an API test

Use the "Call to API Test/Action Dialog Box" on page 1075 (described on page 1075), which you open as follows:

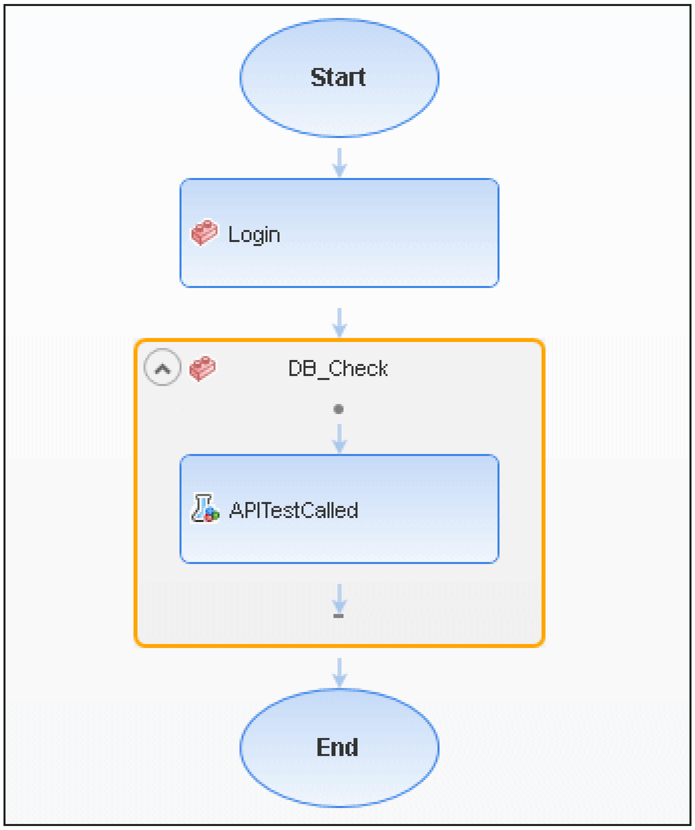
- To insert a new call, select **Design > Call to Existing API Test/Action** or click the **Insert Call to New Action** toolbar button and select **Call to Existing API Test/Action**.
- To modify an existing call, right-click the step in the canvas or the Editor and select **Edit Call to API Test/Action**.

> **Note:** Do not insert a call to an API test or action that contains a call to a GUI test, as this can cause unexpected behavior.

UFT inserts a step that calls the API test or action, for example:

```
RunAPITest "<API test name>"
```

The call to the API test is displayed in the canvas as a call inside the relevant action of the GUI test:



3. Use value of the API test parameters in a GUI test

After you add a call to an API test or action, the input and output parameters are available for use in your GUI test.

The use of parameter values contained in an API test during a GUI test run differs if you are using input and output parameters:

- If you want to use values for the API test **input** parameters during a GUI test run, specify the parameter value in the "Call to API Test/Action Dialog Box". When the API test is run during the GUI test, UFT uses the values you specified for the appropriate parameters in the API test.

- If you want to use values for the API test **output** parameters, you must assign a variable name to the output parameter in the "Call to API Test/Action Dialog Box" or assign the API test output value to a variable or a data table parameter in the GUI test. This variable or data table parameter is then available to use in other steps of the GUI test.

For details on using parameters in your GUI test or action, see "Parameterizing Object Values" on page 1526.

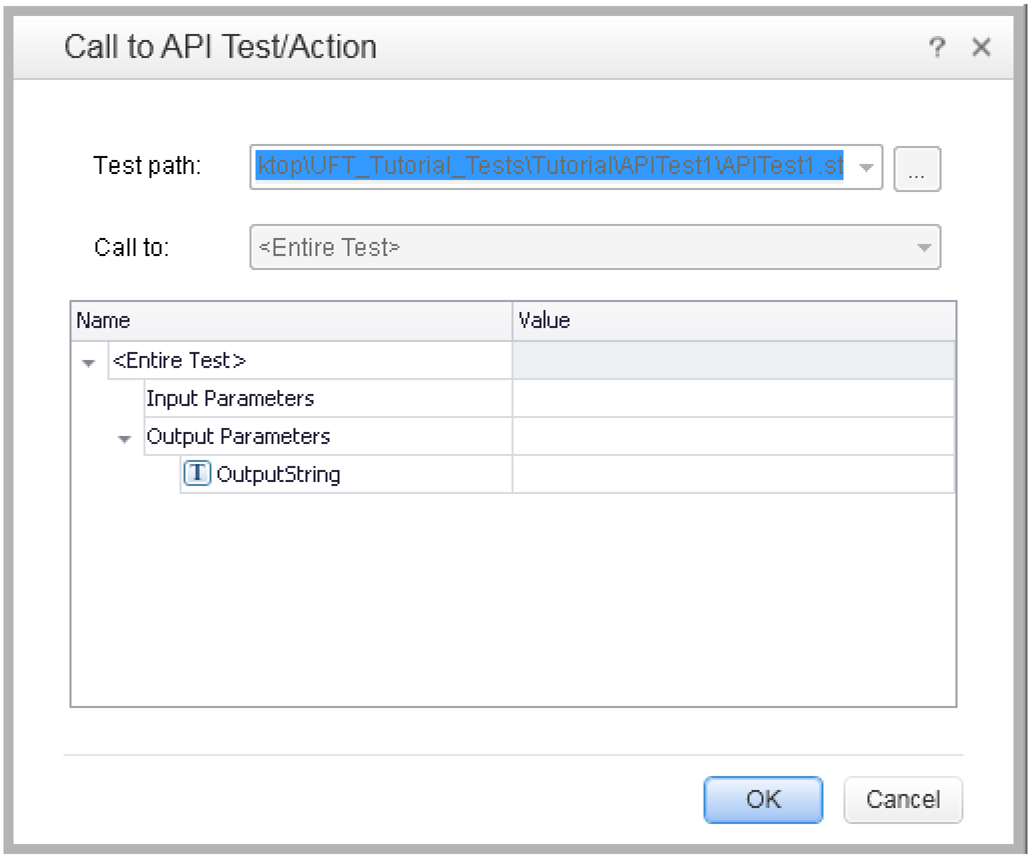
Reference

Call to API Test/Action Dialog Box

Relevant for: GUI tests only

This dialog box enables you to specify the test path, the context (entire test or specific action), and parameter values for a call to an API test or action.

The following image shows an example of a call to an API test with one argument for which you can defined a value.



Test path: ktop\UFT_Tutorial_Tests\Tutorial\APITest1\APITest1.st

Call to: <Entire Test>

Name	Value
<Entire Test>	
Input Parameters	
Output Parameters	
OutputString	

To access	1. Do one of the following: ▪ Ensure that a GUI test or action is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or action node. 2. Do one of the following: ▪ **New call:** Select **Design > Call to Existing API Test/Action** or click the **Insert Call to New Action** toolbar button and select **Call to Existing API Test/Action**. ▪ **Existing call:** Right-click the step in the canvas or the editor and select **Edit Call to API Test/Action**.
Important information	• If you double click on the call to the test or action in the **External Tests** node in the Solution Explorer, the test is opened in the document pane and added to the current solution. • Do not insert a call to an API test that contains a call to a GUI test, as this can cause unexpected behavior.
Relevant tasks	"How to Integrate API Tests and Actions in GUI Tests" on page 1073
See also	"Integrating API Tests and Actions into GUI Tests" on page 1068

User interface elements are described below:

UI Elements	Description
Test path	The path to the API test to which you are inserting a call. You can insert a test path in any of the following ways: • browse to a test path • select a test path from the drop down list of recently called API tests • enter a test path manually You can insert a relative or absolute test path.
Call to	The API test or action to which you are inserting a call. You can insert a call to: • the entire test • a specific action in the test

UI Elements	Description
<Argument grid>	The list of arguments in the API test, including both input and output parameters. If an argument has a default value, that value is displayed in the **Value** column in the Keyword View of the action in which the test is called. You can enter a constant value for each argument, or you can use parameter values from a data table, random number parameter, environment variable parameter, or a GUI test or action input parameter. You insert a parameter value by clicking the parameterization button <#> and specifying it in the "Value Configuration Options Dialog Box" (described on page 1579). **Note:** • Parameter values are read-only in a **Values** cell. To modify a parameter value, click the **X** to clear the cell. Then enter a replacement value. • XML structures are not relevant for this option because they are not supported as argument types for API tests and actions.

Chapter 39: Maintaining and Updating GUI Tests or Components

Relevant for: GUI tests and components

This chapter includes:

Concepts

Why Tests or Components Fail

Relevant for: GUI tests and components

Tests or components fail when UFT encounters a step it cannot perform or the results of a step indicate failure. In many cases this is due to the application being tested not functioning properly. UFT then provides you with run results that assist you in understanding how to fix your application.

Sometimes a test or component fails because the application being tested has changed from when the test or component was created and the test or component needs to be updated to reflect those changes. Your object repository may also be missing some of the objects it needs to run the test. UFT provides tools that help identify and resolve some of these issues.

Object Changes

When UFT runs a step in a test or component, it looks for the object referred to by that step, in the object repositories associated with that test or component. Using the description of the object in the repository, UFT attempts to identify that object in the application.

UFT may not be able to identify the object in the application for a number of reasons:

The Object Does Not Exist in the Application

UFT cannot find an object in the application that matches the description of the object in the object repository. The Maintenance Run Wizard enables you to identify the object that you want your test or component to use.

The Parent Object Changed

UFT cannot find an object in the application that matches and has the same hierarchy as the object in the object repository. The Maintenance Run Wizard enables you to identify the object that you want your test or component to use.

The Object Description Property Values Changed

UFT cannot find an object in the application that is similar to, and has the same description property values as the object in the object repository. The Maintenance Run Wizard enables you to identify the object that you want your test or component to use.

The Object Does Not Exist in the Object Repository

UFT looks for the object to which the test or component refers, in the associated object repositories before attempting to identify that object in the application. If the object in your test or component cannot be found in any associated object repository, The Maintenance Run Wizard enables you to identify the object in your application that you want to add to your repository and use in your test or component.

Checkpoint Changes

Checkpoints fail when they encounter conditions in the application being tested that are unexpected. In many cases this is due to the application not functioning properly. UFT provides you with run results that assist you in understanding how to fix your application.

Sometimes checkpoints fail because the application has changed since the test or component was created and the checkpoints need to be updated to reflect those changes. Update Run Mode enables you to update the checkpoints in your test or component to reflect changes in the application.

For example, suppose your application has an edit box whose default value used to be <Enter value> and you have a checkpoint that checks this value before a new value is entered in the edit box. If the default value in the application changes to be <Enter name> then your checkpoint will fail. Update Run Mode enables you to update the expected values of your checkpoint to reflect the change in the application.

Maintenance Run Mode

Relevant for: GUI tests and components

You can use Maintenance Run Mode to update the test objects in the object repositories associated with your test or component when UFT cannot locate one or more objects in your application during a run session.

When you run a test or component in Maintenance Run Mode, the Maintenance Run Wizard opens each time it encounters any of following problems and provides the described solutions:

Problem	Solution
The step failed because the object in your test or component cannot be identified in the application.	The Maintenance Run Wizard helps you identify the object in the application that you want your test or component to use. If you point to an object in the application being tested, the Maintenance Run Wizard compares that object to the objects in the associated object repositories. Depending on how the property values of the object to which you point compare to the property values of the objects in the associated repositories, the Maintenance Run Wizard suggests one of several options for updating your test or component to reflect the changes in the application. You can also choose to add a comment to your test or component before the failed step.

Problem	Solution
The step failed because the object in your test or component is missing from your associated object repositories.	The Maintenance Run Wizard helps you add the missing object to the repository. You can also choose to add a comment to your test or component before the failed step.
The object in your step exists in the application, but can be identified only through Smart Identification.	Identifying objects using Smart Identification may cause tests or components to run slower. (For details, see "Smart Identification" on page 1316.) The Maintenance Run Wizard helps you modify the description of the object, so that Smart Identification is not needed.

When the Maintenance Run Mode ends, the Maintenance Run Wizard provides a summary of the changes it made to your test or component. The Run Results Viewer also contains a Maintenance Summary that displays details of the changes made to your test or component, including updated and added objects, updated and commented steps, and a summary of changes to the object repository.

Note: Maintenance Run Mode does not support complex checkpoint types such as File checkpoints and XML checkpoints. During the Maintenance Run, these checkpoints run as they would in a regular run session and will fail if there are differences between expected and actual values.
(This note applies to output values as well as checkpoints.)

Update Run Mode

Relevant for: GUI tests and components

You can use Update Run Mode to update your test object descriptions, checkpoints, output values, or, for tests and scripted components, Active Screen captures. When you run a test or component in Update Run Mode, UFT runs the test or component to update the set of identification properties used for test object descriptions, the Active Screen images and values, and/or the expected checkpoint values.

UFT updates the set of identification properties for each object class in your associated object repositories according to the properties currently defined in the "Object Identification Dialog Box" (described on page 1325). If the case-sensitivity designation of a property value has changed since you learned the object, update run mode also updates that setting. After you save the test or component, the updated data is used for subsequent runs.

For task details, see "How to Update Test Object Descriptions, Checkpoints, or Output Values, or Active Screen Captures" on page 1088.

How Test Object Descriptions are Updated when UFT uses Smart Identification

If you have a test or component that runs successfully, but in which certain objects are identified using Smart Identification, you can change the set of properties used for object identification and then use the **Update test object descriptions** option to update the test object description to use the set of properties that Smart Identification used to identify the object.

When you run the test or component with **Update test object descriptions** selected, UFT finds the test object specified in each step based on its current test object description. If UFT cannot find the test object based on its description, it uses the Smart Identification properties to identify the test object (if Smart Identification is enabled). After UFT finds the test object, it then updates its description based on the mandatory and assistive properties that you define in the "Object Identification Dialog Box" (described on page 1325).

How Test Object Descriptions are Updated When they Contain Parameters or Regular Expressions

Any properties that were used in the previous test object description and are no longer part of the description for that test object class, as defined in the "Object Identification Dialog Box" (described on page 1325), are removed from the new description, even if the values were parameterized or defined as regular expressions.

If the same property appears both in the test object's new and previous descriptions, one of the following occurs:

- If the property value in the previous description is parameterized or specified as a regular expression and matches the new property value, UFT keeps the property's previous parameterized or regular expression value. For example, if the previous property value was defined as the regular expression button.*, and the new value is button1, the property value remains button.*.

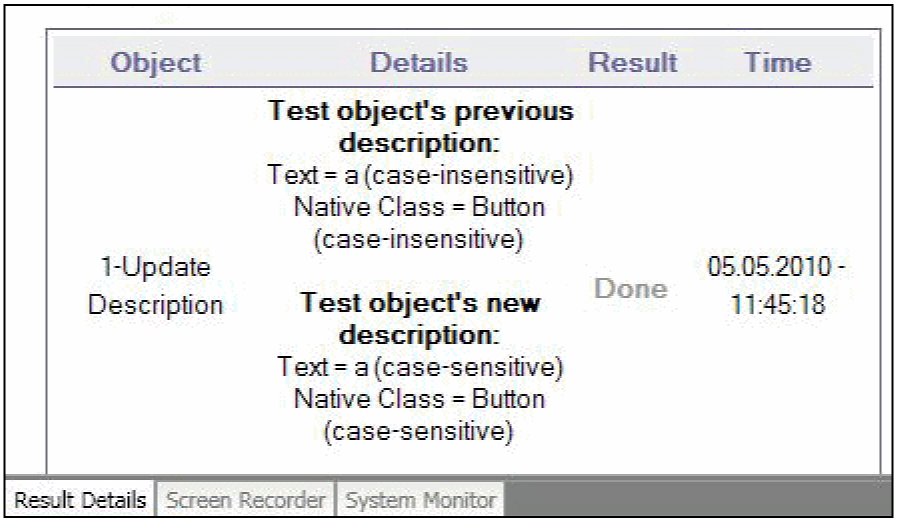
- If the property value in the previous description does not match the new property value, but the object is found using Smart Identification, UFT updates the property value to the new, constant property value. For example, if the previous property value was button.*, and the new value is My button, if a Smart Identification definition enabled UFT to find the object, My button becomes the new property value. In this case, any parameterization or use of regular expressions is removed from the test object description.

How the Case-Sensitivity of Properties is Updated

Most identification property values are not case-sensitive, meaning that UFT identifies an object in an application if it matches the test object description, even if the capitalization of the property values is different. However, UFT does treat some identification property values for specific test objects as case-sensitive. In some cases, the case-sensitivity of some identification properties may change from one version of UFT to the next, or as a result of a patch or hotfix installation.

In those cases, when you use Update Run to update test object descriptions, UFT also updates any case-sensitivity settings that may have changed.

Below is an example of the run results for a step in which the case-sensitivity of some identification properties were updated from **case-insensitive** to **case-senstive**.:



Object	Details	Result	Time
1-Update Description	**Test object's previous description:** Text = a (case-insensitive) Native Class = Button (case-insensitive) **Test object's new description:** Text = a (case-sensitive) Native Class = Button (case-sensitive)	Done	05.05.2010 - 11:45:18

Example of When to Update Test Object Descriptions

Updating test object descriptions can be especially useful when you want to create or debug your test or component steps using object property values that are easy to recognize in your application (such as object labels), but may be language- or operating system-dependent. After you debug your test or component, you can use the **Update Run Mode** option to change the object descriptions to use more universal property values.

Suppose you design a test or component for the English version of part of your application. The test objects are recognized according to the identification property values in the English version, some of which may be language-dependent. You now want to use the same test or component for the French version of your application.

To do this, you define properties that are non-language-dependent, so that UFT can use these properties for object identification. For example, you can identify a link object by its target property value instead of its text property value. You can then perform an update run on the English version of this part of your application using these new properties. This modifies the test object descriptions so that you can later run the test or component successfully on the French version of your application.

When to Update Checkpoint Properties and Output Property Values

Suppose you defined a text checkpoint as part of your test, and the text in your application has changed since you created your test. You can update the test to update the checkpoint properties to reflect the new text.

UFT updates the expected values of your checkpoints to reflect any changes that may have occurred in your application since you created the test and updates the list of items that can be retrieved in your output value steps.

Note: Update Run does not support updating complex checkpoint types such as File

> checkpoints and XML checkpoints. During the Update Run, these checkpoints run as they would in a regular run session and will fail if there are differences between expected and actual values.
> (This note applies to output values as well as checkpoints.)

When to Update Active Screen Images and Values (tests and scripted components only)

Suppose a dialog box in your application has changed since you recorded your test. You can update the test to update the dialog box appearance and its properties in the Active Screen.

UFT updates images and property values in the Active Screen to reflect any changes that may have occurred in your application since you recorded the test or if the Active Screen does not appear as it should.

You can also use this option to increase or decrease the amount of information saved and displayed in your Active Screen. Change the Capture Level slider (**Tools > Options > GUI Testing** tab **> Active Screen** node), and run the test in Update Run Mode with the **Update Active Screen images and values** check box selected. UFT updates the amount of information it saves and displays in the Active Screen, based on the new setting. For more details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

The Update Run mode does not update Insight test objects. To update an Insight test object, open the object in the object repository, and click the **Change Test Object Image** button in the **Test object image** area. For more details, see "Change Test Object Image / Add Insight Test Object Dialog Box" on page 1233.

Tasks

How to Use Maintenance Run Mode to Update Your Test or Component When Your Application Changes

Relevant for: GUI tests and components

This task describes how to use the Maintenance Run Wizard to help you maintain your test or component.

This task includes the following steps:

- "Prerequisites" below
- "Decide how long UFT should wait before determining that an object cannot be found (tests only) (Optional)" below
- "Run the test or component in Maintenance Run Mode" on the next page
- "Export and merge changes to your shared object repository (Optional) " on the next page
- "Analyze the results of the Maintenance Run Mode session in the Run Results Viewer" on the next page

1. **Prerequisites**

   - You must have the Microsoft Script Debugger installed to run the tests or components in Maintenance Run Mode. If it is not installed, you can use the UFT Additional Installation Requirements Utility to install it. (Select **Start > Programs > HP Software > HP Unified Functional Testing > Tools > Additional Installation Requirements**.)

     > **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

   - You can run in Maintenance Run Mode only when UFT is set to use the **Normal** (displays execution marker) run mode. It cannot run in **Fast** mode. For details, see "Test Runs Pane (Options Dialog Box > GUI Testing Tab)" on page 539.

   - You cannot run tests or components in Maintenance Run Mode on applications that do not have a user interface.

2. **Decide how long UFT should wait before determining that an object cannot be found (tests only) (Optional)**

   The Maintenance Run Wizard opens often when your application has changed and UFT is unable to identify objects. You may want to decrease the amount of time UFT waits for an object to be displayed before determining that the object cannot be found. You can change the object synchronization timeout in the Run pane of the Test Settings dialog box. Ensure that the timeout specified is sufficient for the objects in your application to load. For more details, see "Run Pane (Test Settings Dialog Box)" on page 601.

After Maintenance Run Mode finishes you may want to return this setting to its previous value for regular test runs. (The default setting is 20 seconds.)

3. **Run the test or component in Maintenance Run Mode**

   a. Select **Run > Maintenance Run Mode** or click the down arrow next to the **Run** button in the toolbar and select **Maintenance Run Mode**.

   b. Specify the results location and the input parameter values (if applicable) for the Maintenance Run Mode session. For details, see "Run Dialog Box" on page 651, and "Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)" on page 655.

   c. Follow the steps in the "Maintenance Run Wizard " on page 1091. For details, see "Maintenance Run Wizard " on page 1091 and"Maintenance Run Wizard Workflow" on the next page.

   > **Tip:** As an alternative to using the Maintenance Run Wizard, you can update individual test object descriptions from the object in your application using the **Update from Application** option in the Object Repository window or Object Repository Manager. For details, see "Updating Identification Properties from an Object in Your Application" on page 1206.

4. **Export and merge changes to your shared object repository (Optional)**

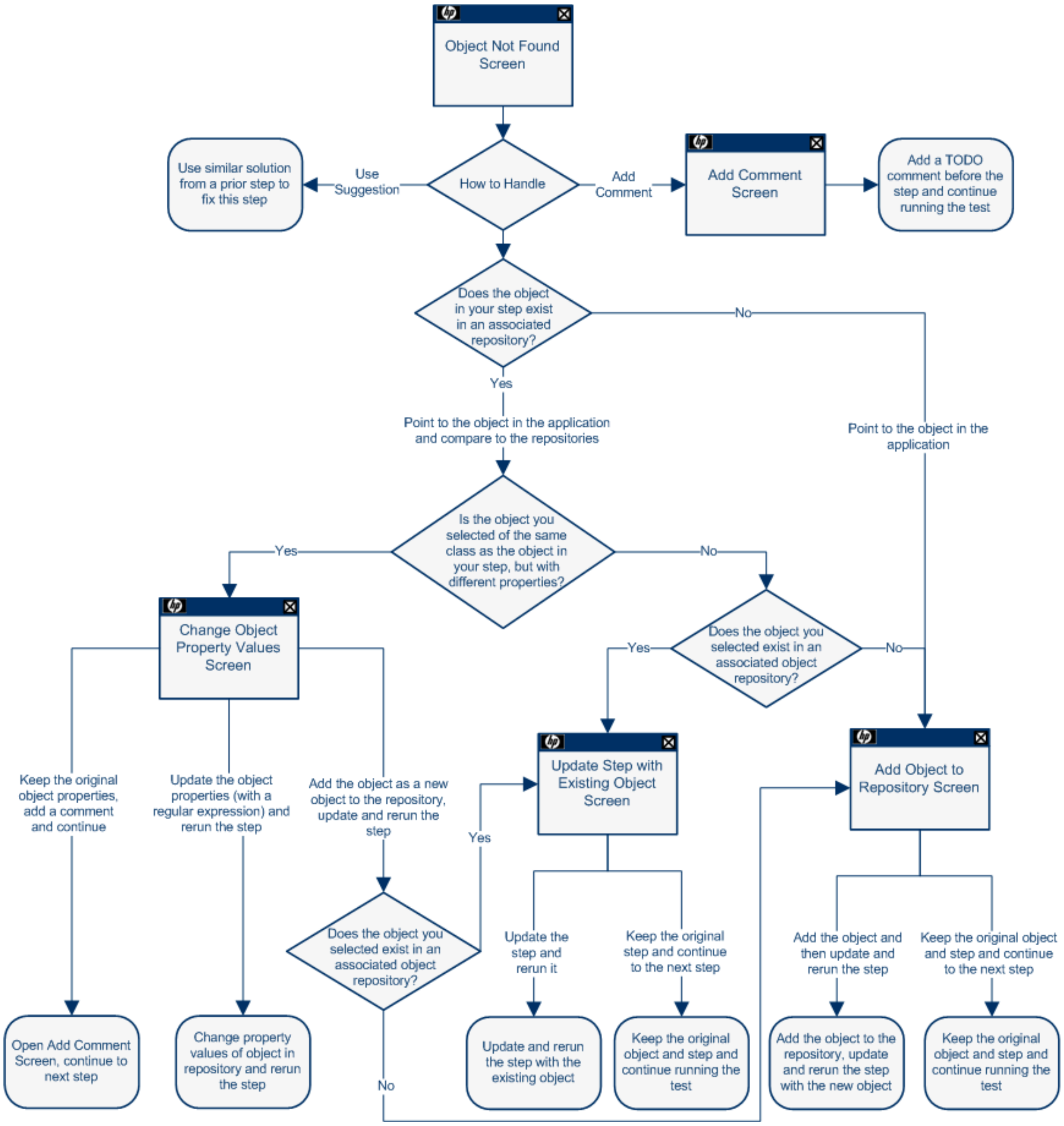
   After using the Maintenance Run Wizard to update the test or component, you may want to use the **Update from Local Repository** option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For details, see "How to Update a Shared Object Repository From a Local Object Repository" on page 1358.

5. **Analyze the results of the Maintenance Run Mode session in the Run Results Viewer**

   By default, when the run session ends, the Run Results Viewer opens. For details on viewing the run session results, see the *HP Run Results Viewer User Guide* If you cleared the **View results when run session ends** check box in the **Run Sessions** pane of the Options dialog box (**Tools > Options > General** tab **> Run Sessions** node), the Run Results Viewer does not open at the end of the run session. For details on the Options dialog box, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 526

Maintenance Run Wizard Workflow

Relevant for: GUI tests and components


Object Not Found Screen
How to Handle
Use Suggestion
Use similar solution from a prior step to fix this step
Add Comment
Add Comment Screen
Add a TODO comment before the step and continue running the test
Does the object in your step exist in an associated repository?
No
Yes
Point to the object in the application and compare to the repositories
Point to the object in the application
Is the object you selected of the same class as the object in your step, but with different properties?
Yes
No
Change Object Property Values Screen
Does the object you selected exist in an associated object repository?
Yes
No
Update Step with Existing Object Screen
Add Object to Repository Screen
Keep the original object properties, add a comment and continue
Update the object properties (with a regular expression) and rerun the step
Add the object as a new object to the repository, update and rerun the step
Yes
Does the object you selected exist in an associated object repository?
No
Update the step and rerun it
Keep the original step and continue to the next step
Add the object and then update and rerun the step
Keep the original object and step and continue to the next step
Open Add Comment Screen, continue to next step
Change property values of object in repository and rerun the step
Update and rerun the step with the existing object
Keep the original object and step and continue running the test
Add the object to the repository, update and rerun the step with the new object
Keep the original object and step and continue running the test

Note: The Object Not Found Page does not open when UFT uses Smart Identification to identify an object in your test. In that case, the Maintenance Run Wizard suggests updating the object properties according to the properties currently defined in the "Object Identification Dialog Box" (described on page 1325).

How to Update Test Object Descriptions, Checkpoints, or Output Values, or Active Screen Captures

Relevant for: GUI tests and components

This task describes how to update your test or component data so that it is accurate for subsequent runs.

This task includes the following steps:

- "Determine which data types you want to update " below
- "Run in Update Run Mode" below
- "Export and merge changes to your shared object repository (Optional) " below
- "Analyze the results of the Update Run Mode in the Run Results Viewer" on the next page

1. **Determine which data types you want to update**

   For details, see "Update Run Mode " on page 1081.

2. **Run in Update Run Mode**

   a. Specify the settings for the update run process. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

   b. Enter any required input parameter values in the Input Parameters tab. For details, see "Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)" on page 655.

      When UFT updates tests, it runs through only one iteration of the test and one iteration of each action in the test, according to the run option selected. For details on actions, see "Actions in GUI Testing" on page 871.

      When a test runs in Update Run Mode, it does not update parameterized values, such as Data pane data and environment variables. For details on parameterized values and environment variables, see "Parameterizing Object Values" on page 1526. Update Run Mode does not modify the property values of existing object descriptions in the object repository. To fix the object property values to match your application, use **Maintenance Run Mode**. For more details, see "How to Use Maintenance Run Mode to Update Your Test or Component When Your Application Changes " on page 1085.

3. **Export and merge changes to your shared object repository (Optional)**

   When UFT updates tests or components, it always saves the updated objects in the local object repository, even if the objects being updated were originally from a shared object repository. The next time you run the tests or components, UFT uses the objects from the local object repository, as the local object repository has a higher priority than any shared object repositories.

After using Update Run Mode to update the test or component, you may want to use the Update from Local Repository option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For details, see "How to Update a Shared Object Repository From a Local Object Repository" on page 1358.

4. **Analyze the results of the Update Run Mode in the Run Results Viewer**

   The run results for an update run session are always saved in a temporary location.

   Test objects that cannot be identified during the update process are not updated. As in any run session, if an object cannot be found during the update run, the run session fails, and information on the failure is included in the Run Results. In these situations, you may want to use **Maintenance Run Mode** to resolve these problems.

   Because Update Run does not support updating complex checkpoint types such as File checkpoints and XML checkpoints, then these checkpoints run as they would in a regular run session and will fail if there are differences between expected and actual values.

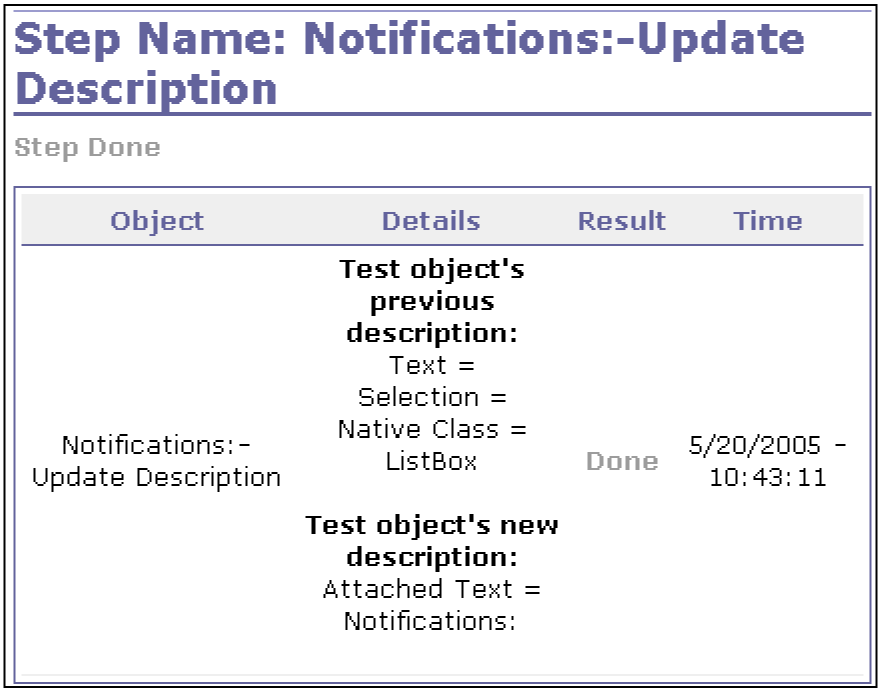
   By default, when the run session ends, the Run Results Viewer opens. For details on viewing the run session results, see the *HP Run Results Viewer User Guide*. If you cleared the **View results when run session ends** check box in the **Run Sessions** pane of the Options dialog box (**Tools > Options > General** tab **> Run Sessions** node), the Run Results Viewer does not open at the end of the run session. For details on the Options dialog box, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 526.

   When the update run ends, the Run Results Viewer can show:

   - Updated values for checkpoints.

   - Updated test object descriptions.

For example:



Step Name: Notifications:-Update Description

Step Done

Object	Details	Result	Time
Notifications:-Update Description	**Test object's previous description:** Text = Selection = Native Class = ListBox **Test object's new description:** Attached Text = Notifications:	Done	5/20/2005 - 10:43:11

Reference

Maintenance Run Wizard

Relevant for: GUI tests and components

This wizard enables you to update your test or component when your application changes.

To access	1. Do one of the following: ▪ Ensure that a GUI test or component is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or component node. 2. Use one of the following: ▪ Select **Run > Maintenance Run Mode** ▪ Click the down arrow next to the **Run** button in the toolbar and select **Maintenance Run Mode**.
Important information	See the prerequisite information in "How to Use Maintenance Run Mode to Update Your Test or Component When Your Application Changes " on page 1085.
Relevant tasks	"How to Use Maintenance Run Mode to Update Your Test or Component When Your Application Changes " on page 1085
Show me	To see a demonstration of this functionality, go to the HP Live Network (HPLN) page and select the **Maintenance Run Mode** movie from the list.
Wizard map	"Maintenance Run Wizard Workflow" on page 1087
See also	• "Why Tests or Components Fail" on page 1079 • "Maintenance Run Mode" on page 1080

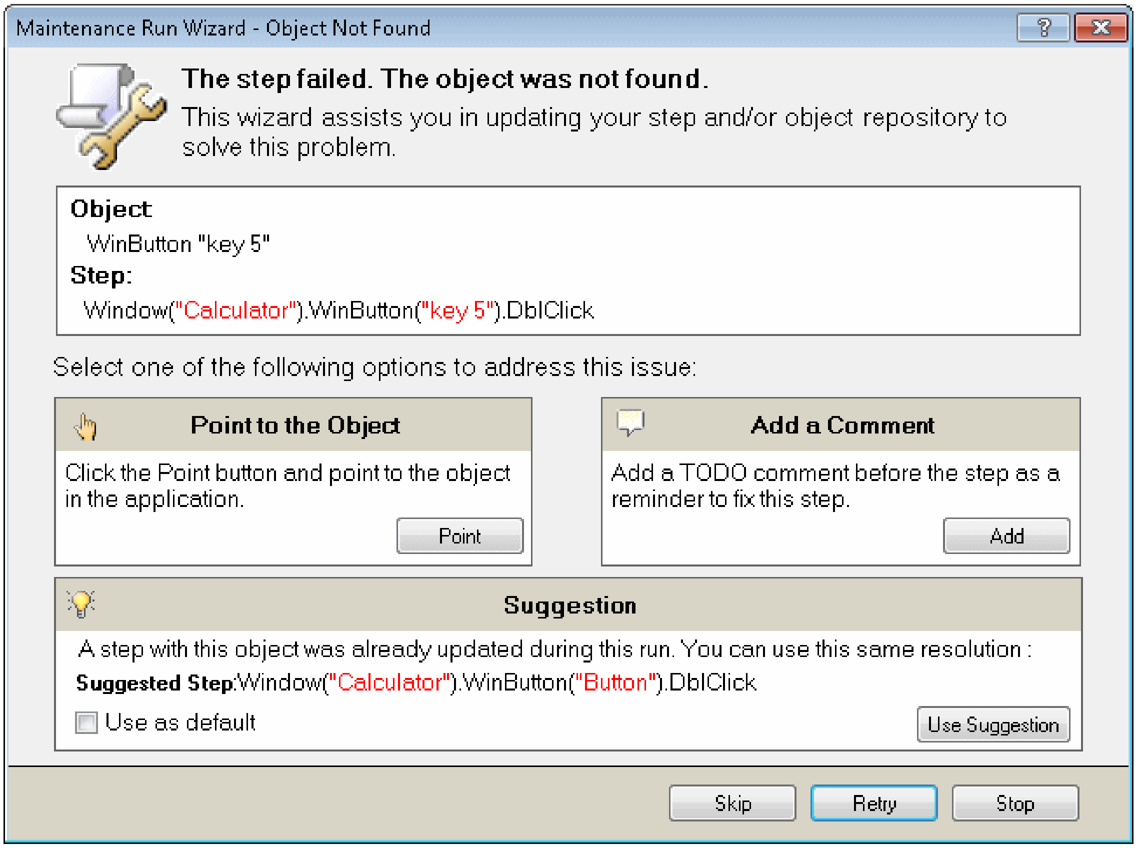
The wizard contains the following pages:

Object Not Found Page (Maintenance Run Wizard)

Relevant for: GUI tests and components

This wizard page enables you to identify the **Object** that could not be found and the **Step**UFT was trying to perform.

The Object Not Found page opens when an object in your test or component cannot be found in the application you are testing or in the associated object repositories.



Maintenance Run Wizard - Object Not Found

The step failed. The object was not found.

This wizard assists you in updating your step and/or object repository to solve this problem.

Object

WinButton "key 5"

Step:

Window("Calculator").WinButton("key 5").DblClick

Select one of the following options to address this issue:

Point to the Object

Click the Point button and point to the object in the application.

Point

Add a Comment

Add a TODO comment before the step as a reminder to fix this step.

Add

Suggestion

A step with this object was already updated during this run. You can use this same resolution :

Suggested Step:Window("Calculator").WinButton("Button").DblClick

□ Use as default

Use Suggestion

Wizard map	"Maintenance Run Wizard Workflow" on page 1087

See also	The wizard contains the following additional pages: - "Add Comment Page (Maintenance Run Wizard)" on the next page - "Change Object Property Values Page (Maintenance Run Wizard)" on page 1096 - "Update Step with Existing Object Page (Maintenance Run Wizard)" on page 1099 - "Add Object to Repository Page (Maintenance Run Wizard)" on page 1101 - "Smart Identification Page (Maintenance Run Wizard)" on page 1104 - "Maintenance Mode Summary Page (Maintenance Run Wizard)" on page 1106

User interface elements are described below:

UI Elements	Description
Point to the Object	Click the **Point** button and point to the object in the application that should be used in the step. Use this option if you know the application has changed and identifying a new object for use in the step will resolve the issue, or if the object does not exist in the associated object repositories. If the location to which you point is associated with several objects, the "Object Selection Dialog Box " (described on page 1195) opens. Select the correct object from the tree and click **OK**. The Point to the Object option is disabled when: - The test or component is open in read-only mode. - The object is used within a function library function. - The object's method is defined as a registered user function.
Add a Comment	Use this option if you want to add a comment to your test as a reminder to fix the failed step. The Add a Comment option is disabled when: - The test or component is open in read-only mode. - The object is used within a function library function. - The object's method is defined as a registered user function.

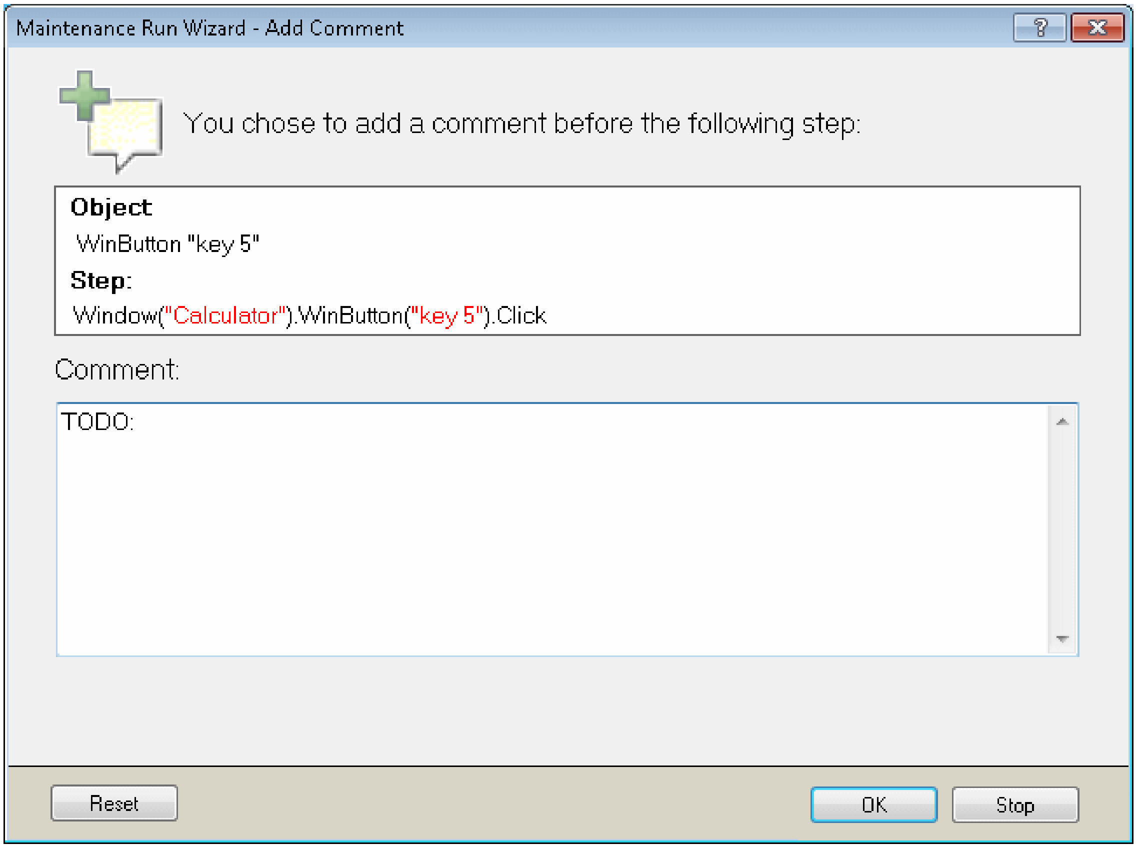
UI Elements	Description
Suggestion	Displayed only if the Maintenance Run Wizard cannot find an object in the application that was not found earlier in the Maintenance Run Wizard run as well. If, when the object was first not found, you chose to replace it with a different object, the Maintenance Run Wizard suggests replacing it with the same object now.
Use as default	If, in subsequent steps the same object cannot be found, the Maintenance Run Wizard automatically replaces the object not found with the object you added to the object repository. The Maintenance Run Wizard does not open on these subsequent steps.
Skip	Skips the current step in the test or component and continues to run the Maintenance Run Wizard on the remainder of the test or component. This can be used when the problem is in the application being tested and not in the test or component. Before clicking Skip, ensure that the application is ready for the next step in the test or component.
Retry	Retries the current step.
Stop	Stops the Maintenance Run and opens the "Maintenance Mode Summary Page (Maintenance Run Wizard)".

Add Comment Page (Maintenance Run Wizard)

Relevant for: GUI tests and components

This wizard page enables you to add a comment to your test or component before the current step. This can be used when you believe there is a problem in your test or component, but identifying the object in the application will not solve the problem, or you want to fix the test or component manually.

The Add Comment page creates a comment in your test or component beginning with the word TODO along with text you add, as a reminder to fix the step at a later time.



You chose to add a comment before the following step:

Object

WinButton "key 5"

Step:

```
Window("Calculator").WinButton("key 5").Click
```

Comment:

TODO:

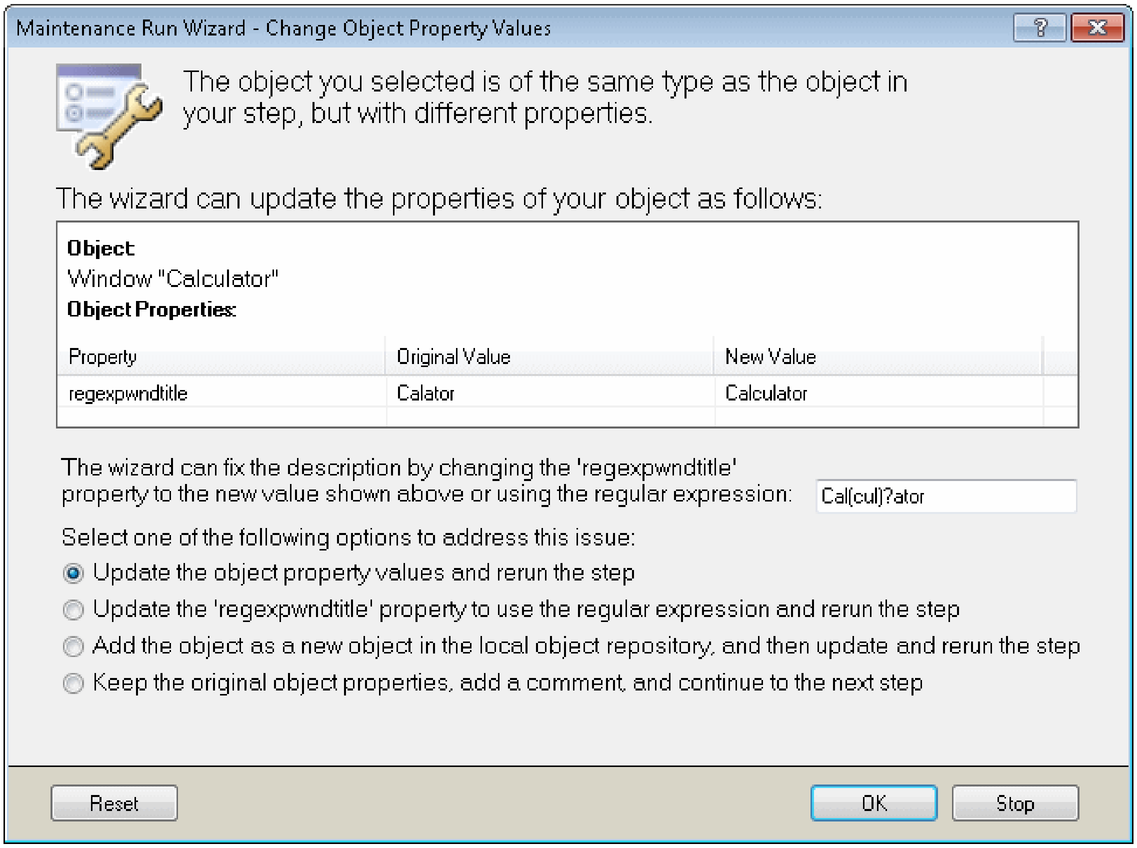
Wizard map	"Maintenance Run Wizard Workflow" on page 1087
See also	The wizard contains the following additional pages: • "Object Not Found Page (Maintenance Run Wizard)" on page 1092 • "Change Object Property Values Page (Maintenance Run Wizard)" on the next page • "Update Step with Existing Object Page (Maintenance Run Wizard)" on page 1099 • "Add Object to Repository Page (Maintenance Run Wizard)" on page 1101 • "Smart Identification Page (Maintenance Run Wizard)" on page 1104 • "Maintenance Mode Summary Page (Maintenance Run Wizard)" on page 1106

Change Object Property Values Page (Maintenance Run Wizard)

Relevant for: GUI tests and components

This wizard page enables you to update the property values of the object in the associated object repository to match the property values of the object to which you pointed in the application.

The Change Object Property Values page opens when the object to which you pointed is of the same class as the object in your step, but it has different description property values.



Maintenance Run Wizard - Change Object Property Values

The object you selected is of the same type as the object in your step, but with different properties.

The wizard can update the properties of your object as follows:

Object
Window "Calculator"
Object Properties:

Property	Original Value	New Value
regexpwndtitle	Calator	Calculator

The wizard can fix the description by changing the 'regexpwndtitle' property to the new value shown above or using the regular expression: Cal(cul)?ator

Select one of the following options to address this issue:

- ● Update the object property values and rerun the step
- ○ Update the 'regexpwndtitle' property to use the regular expression and rerun the step
- ○ Add the object as a new object in the local object repository, and then update and rerun the step
- ○ Keep the original object properties, add a comment, and continue to the next step

Important information	If the object to which you point has a different parent object than the one in the object repository and has different property values, the Change Object Property Values page opens twice. The first time it enables you to update the parent object of the object in the object repository to match the parent object of the object to which you pointed. The second time it enables you to update the object in the object repository to match the object to which you pointed.
Wizard map	"Maintenance Run Wizard Workflow" on page 1087

See also	The wizard contains the following additional pages: - "Object Not Found Page (Maintenance Run Wizard)" on page 1092 - "Add Comment Page (Maintenance Run Wizard)" on page 1094 - "Update Step with Existing Object Page (Maintenance Run Wizard)" on page 1099 - "Add Object to Repository Page (Maintenance Run Wizard)" on page 1101 - "Smart Identification Page (Maintenance Run Wizard)" on page 1104 - "Maintenance Mode Summary Page (Maintenance Run Wizard)" on page 1106

Change Object Property Values Page - Central Area

User interface elements of the central area of the Change Object Property Values page are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Object	The object in an associated object repository that is of the same class as the object to which you pointed in the application.
Object Properties	A table displaying the changes that will be made to the property values of the object in the object repository.
Property	The name of the property whose value will be changed.
Original Value	The original property value of the object in the object repository.
New Value	The new property value for the object in the object repository, based on the object to which you pointed in the application.

UI Elements	Description
<Recommended regular expression>	Depending on the object to which you pointed, the Change Object Property Value page may include a message that a regular expression can be used to update the property value of the object in the associated object repository. You can modify the suggested regular expression in the edit box. For details on regular expressions, see "Regular Expressions Overview" on page 318. **Note:** • If the Maintenance Run Wizard does not determine that a regular expression is relevant for the new property value, the Change Object Property Value page does not display the suggested regular expression below the properties table. The Update the <property name> property to use the regular expression and rerun the step radio button is also not displayed. • In a situation where more than one property can use a regular expression, the Maintenance Run Wizard suggests a regular expression only for the first property value.

Change Object Property Values Page - Options

User interface elements are described below:

UI Elements	Description
Update the object property and rerun the step	Updates the property values of the object in the object repository to match those of the object to which you pointed in the application, and reruns the step. The new property values are shown under New Value.
Update the <property name> property to use the regular expression and rerun the step	Displayed only if the property value can be updated to use a regular expression. Updates the property value of the object in the object repository with the regular expression as shown in the edit box, and reruns the step.
Add the object as a new object in the local object repository, and then update and rerun the step	Adds the object to which you pointed, with its current properties, as a new object in the local object repository. This new object may already exist in an associated object repository. Depending on whether the object you want to add already exists in an associated object repository, one of the following pages opens: • "Update Step with Existing Object Page (Maintenance Run Wizard)" on the next page • "Add Object to Repository Page (Maintenance Run Wizard)" on page 1101

UI Elements	Description
Keep the original object properties, add a comment, and continue to the next step	Keeps the original object properties of the object in the object repository. Opens the Add Comment page, enabling you to add a comment before the step, and then continues to the next step.
Reset	Returns to the Object Not Found page, where you can point to a different object in the application or choose a different course of action for this step.

Update Step with Existing Object Page (Maintenance Run Wizard)

Relevant for: GUI tests and components

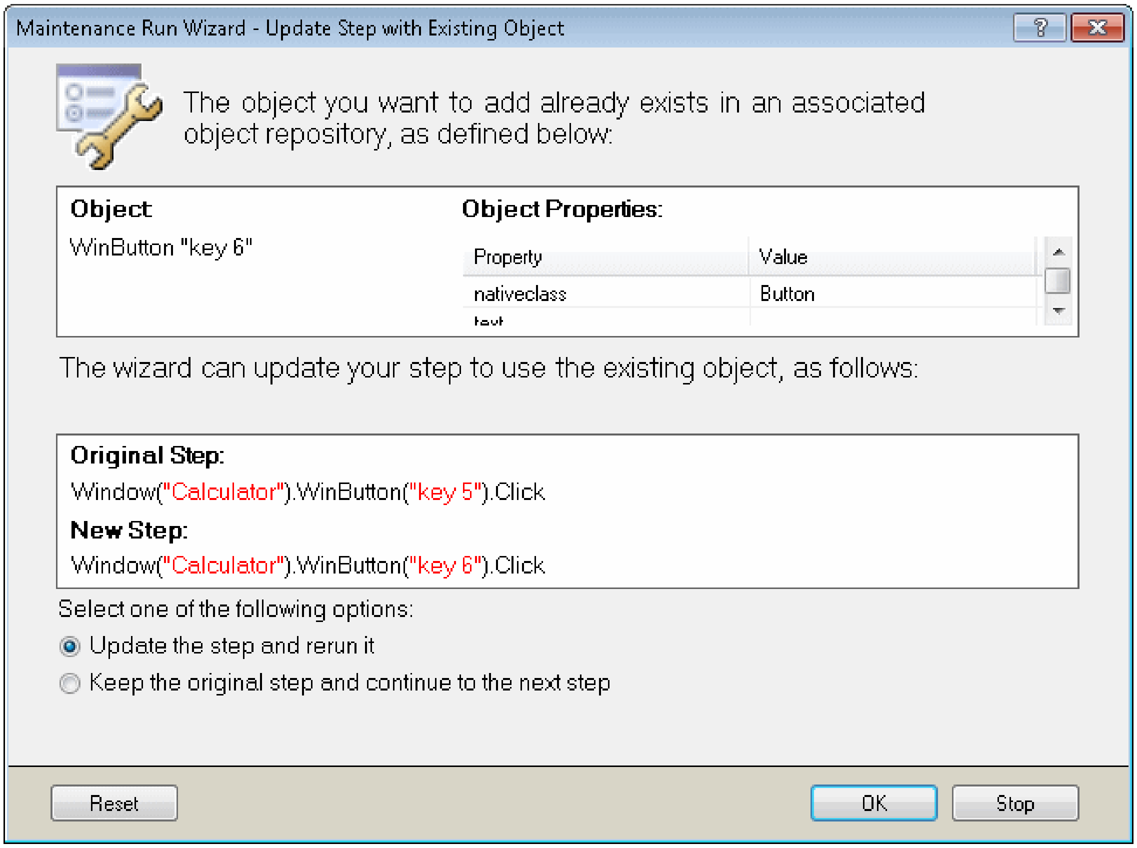
This wizard page enables you to update the step in your test to use an object that already exists in an associated object repository.

The Update Step with Existing Object page opens if the object to which you pointed in the Object Not Found page exists in an associated object repository and:

- The object to which you pointed is not of the same class as the object in your step, but with different description property values.

Or

- In the Change Object Property Values page you chose **Add the object as a new object in the local object repository, and then update and rerun the step**.



The object you want to add already exists in an associated object repository, as defined below:

Object

WinButton "key 6"

Object Properties:

Property	Value
nativeclass	Button
text	

The wizard can update your step to use the existing object, as follows:

Original Step:

Window("Calculator").WinButton("key 5").Click

New Step:

Window("Calculator").WinButton("key 6").Click

Select one of the following options:

● Update the step and rerun it

○ Keep the original step and continue to the next step

Wizard map	"Maintenance Run Wizard Workflow" on page 1087
See also	The wizard contains the following additional pages: • "Object Not Found Page (Maintenance Run Wizard)" on page 1092 • "Add Comment Page (Maintenance Run Wizard)" on page 1094 • "Change Object Property Values Page (Maintenance Run Wizard)" on page 1096 • "Add Object to Repository Page (Maintenance Run Wizard)" on the next page • "Smart Identification Page (Maintenance Run Wizard)" on page 1104 • "Maintenance Mode Summary Page (Maintenance Run Wizard)" on page 1106

Update Step with Existing Object Page - Central Area

User interface elements of the central area of the Update Step with Existing Object page are described below:

UI Elements	Description
Object	The object in an associated object repository that is the same as the object to which you pointed in the application.
Object Properties	The properties and property values of the object to which you pointed in the test application.
Original Step	The failed original step, with the object that could not be found.
New Step	The new step as it would appear updated to refer to the object which already exists in an associated object repository.

Update Step with Existing Object Page - Options

User interface elements are described below:

UI Elements	Description
Update the step and rerun it	Updates the failed step as shown under New Step and reruns the step. The Maintenance Run Wizard does not remove the original step from your test or component. The original step is converted into a comment and the updated step is added below it.
Keep the original step and continue to the next step	Keeps the original step and continues to run the Maintenance Run Wizard on the remainder of the test or component.
Reset	Returns to the Object Not Found page, where you can point to a different object in the application or choose a different course of action for this step.

Add Object to Repository Page (Maintenance Run Wizard)

Relevant for: GUI tests and components

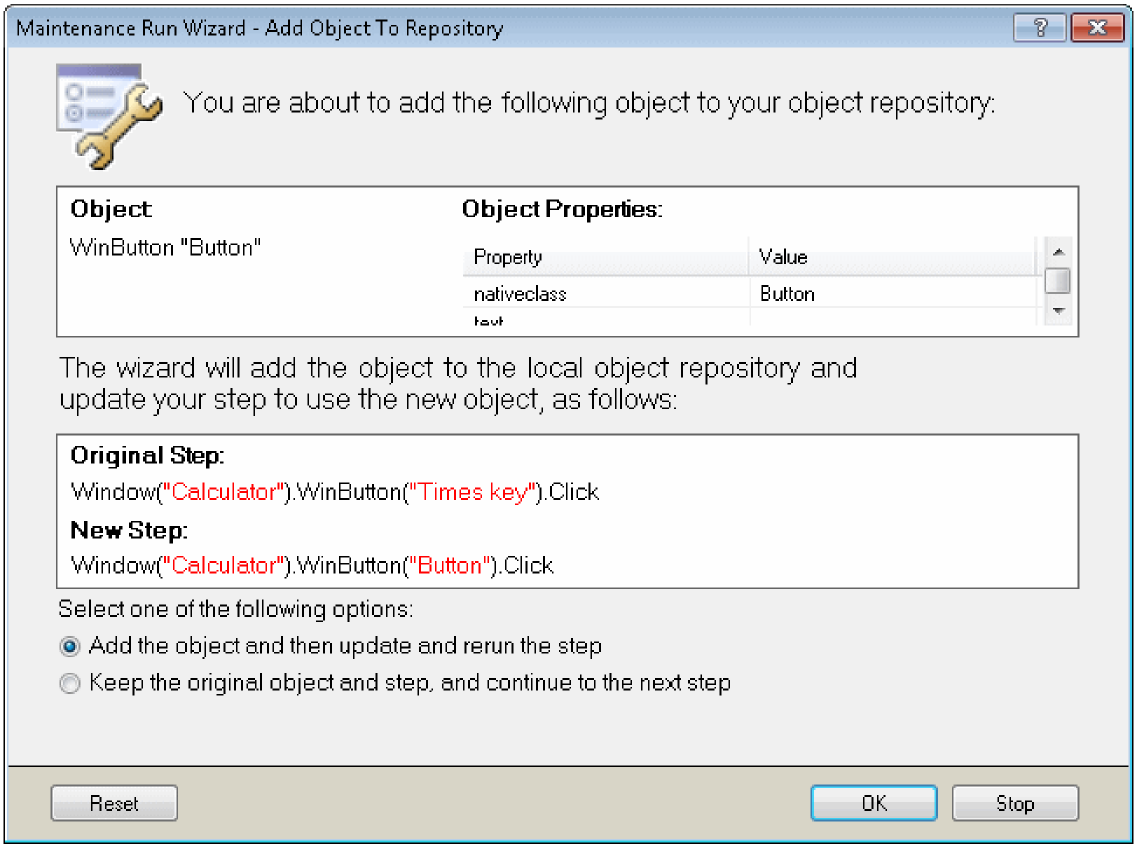
This wizard page enables you to add the object to which you pointed to the object repository.

The Add Object to Repository page opens when:

- the object **in your step** does not exist in any associated repository.
- the object **to which you pointed** does not exist in any associated object repository and:
  - the object to which you pointed is not of the same class as the object in your step, but with different description property values.

  Or

  - in the Change Object Property Values page you chose **Add the object as a new object in the local object repository, and then update and rerun the step**.



You are about to add the following object to your object repository:

Object

WinButton "Button"

Object Properties:

Property	Value
nativeclass	Button
text	

The wizard will add the object to the local object repository and update your step to use the new object, as follows:

Original Step:

```
Window("Calculator").WinButton("Times key").Click
```

New Step:

```
Window("Calculator").WinButton("Button").Click
```

Select one of the following options:

● Add the object and then update and rerun the step

○ Keep the original object and step, and continue to the next step

Wizard map	"Maintenance Run Wizard Workflow" on page 1087

See also	The wizard contains the following additional pages: • "Object Not Found Page (Maintenance Run Wizard)" on page 1092 • "Add Comment Page (Maintenance Run Wizard)" on page 1094 • "Change Object Property Values Page (Maintenance Run Wizard)" on page 1096 • "Update Step with Existing Object Page (Maintenance Run Wizard)" on page 1099 • "Smart Identification Page (Maintenance Run Wizard)" on the next page • "Maintenance Mode Summary Page (Maintenance Run Wizard)" on page 1106

Add Object to Repository Page - Central Area

User interface elements are described below:

UI Elements	Description
Object	The object to which you pointed in the test application.
Object Properties	The properties and property values of the object to which you pointed in the test application.
Original Step	The failed original step, with the object that could not be found.
New Step	The new step as it would appear updated to refer to the object being added to the object repository.

Add Object to Repository Page - Options

User interface elements are described below:

UI Elements	Description
Add the object and then update and rerun the step	Adds the new object to the object repository, updates the failed step as shown under New Step and reruns the step. The Maintenance Run Wizard does not remove the original step from your test or component. The original step is converted into a comment and the updated step is added below it.
Keep the original object and step, and continue to the next step	Keeps the original step containing the original object and continues to run the Maintenance Run Wizard on the remainder of the test or component.
Reset	Returns to the Object Not Found page, where you can point to a different object in the application or choose a different course of action for this step.

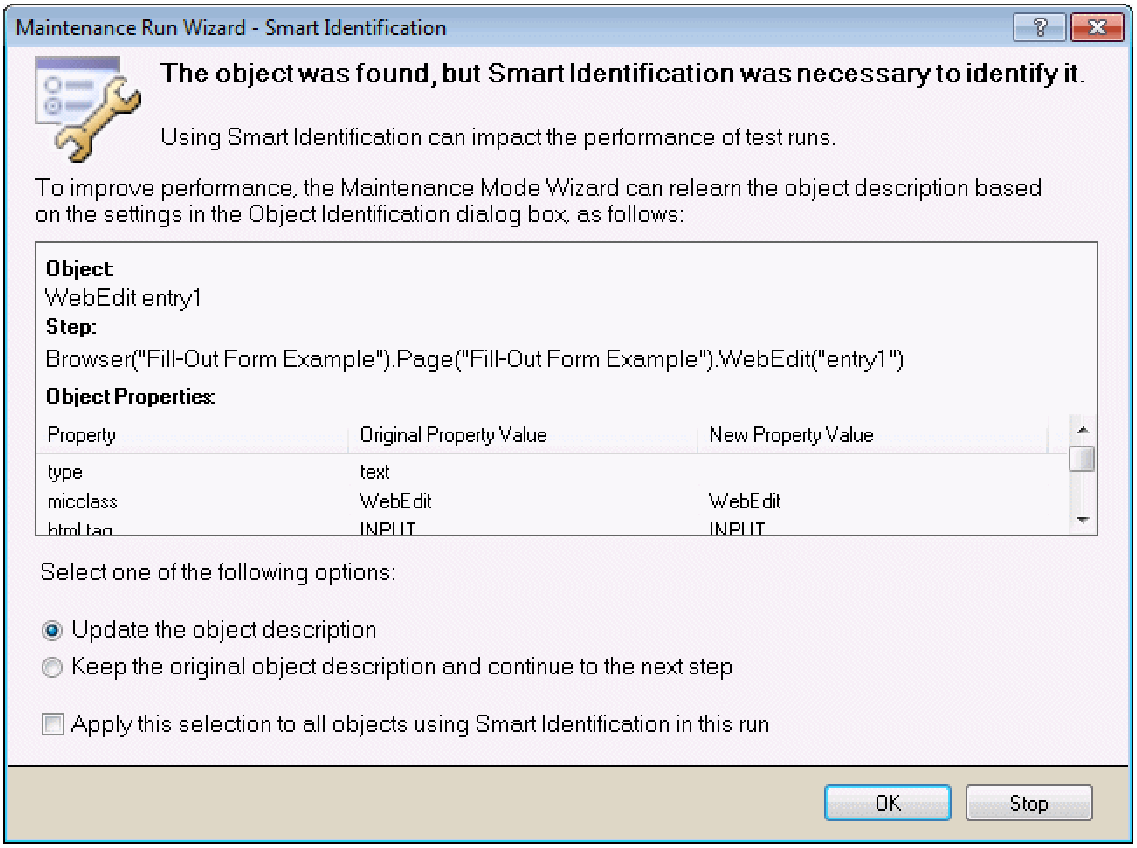
Smart Identification Page (Maintenance Run Wizard)

Relevant for: GUI tests and components

This wizard page enables you to update the object description according to the properties currently defined in the "Object Identification Dialog Box" (described on page 1325).

The Smart Identification page opens if UFT used the Smart Identification mechanism to identify the object in your test or component. For details on the Smart Identification mechanism, see "Smart Identification" on page 1316.

Smart Identification may slow down test or component execution, as it is only activated after the object synchronization timeout has been reached.



The object was found, but Smart Identification was necessary to identify it.

Using Smart Identification can impact the performance of test runs.

To improve performance, the Maintenance Mode Wizard can relearn the object description based on the settings in the Object Identification dialog box, as follows:

Object:
WebEdit entry1
Step:
Browser("Fill-Out Form Example").Page("Fill-Out Form Example").WebEdit("entry1")
Object Properties:

Property	Original Property Value	New Property Value
type	text	
micclass	WebEdit	WebEdit
html tag	INPUT	INPUT

Select one of the following options:

● Update the object description
○ Keep the original object description and continue to the next step

□ Apply this selection to all objects using Smart Identification in this run

Important information	Your test may use Smart Identification because the set of properties used for object identification does not uniquely identify all the objects of a particular class. You can use the **Update test object descriptions** option of **Update Run Mode** to change the set of properties used for object identification for all objects in a specific class. This may be more useful than using the Maintenance Run Wizard for every object in that class. For more details, see "How Test Object Descriptions are Updated when UFT uses Smart Identification" on page 1081.

Wizard map	"Maintenance Run Wizard Workflow" on page 1087
See also	The wizard contains the following additional pages: • "Object Not Found Page (Maintenance Run Wizard)" on page 1092 • "Add Comment Page (Maintenance Run Wizard)" on page 1094 • "Change Object Property Values Page (Maintenance Run Wizard)" on page 1096 • "Update Step with Existing Object Page (Maintenance Run Wizard)" on page 1099 • "Add Object to Repository Page (Maintenance Run Wizard)" on page 1101 • "Maintenance Mode Summary Page (Maintenance Run Wizard)" on the next page

Smart Identification Page - Central Area

User interface elements are described below:

UI Elements	Description
Object	The object in your application that required the use of the Smart Identification mechanism to be identified.
Step	The step in your test in which the object is referenced.
Object Properties	**Property.** The list of properties in the old and new object description. **Original Property Value.** The original value of the property in the **Property** column. Properties that have no value were not part of the original object description. **New Property Value.** The new value of the property in the **Property** column.

Smart Identification Page - Options

User interface elements are described below:

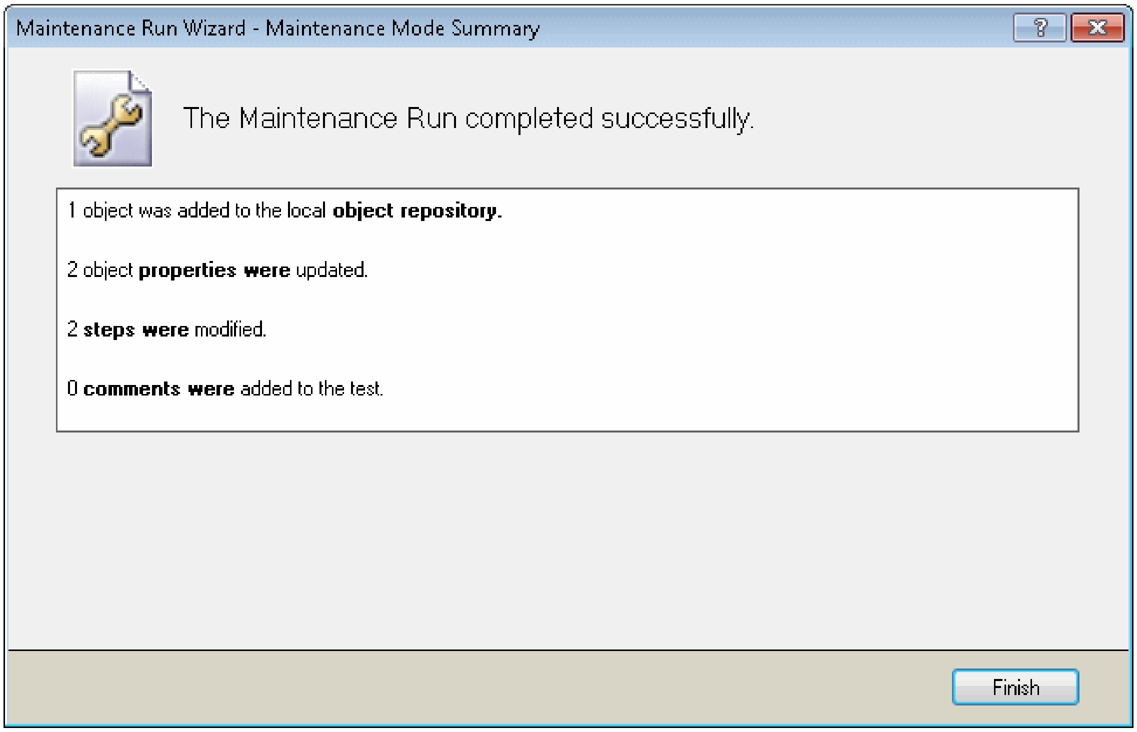
UI Elements	Description
Update the object description	Updates the object description to use the set of properties currently defined in the "Object Identification Dialog Box" on page 1325 for the object in your test. Make sure that the set of properties defined in the Object Identification dialog box for the object is sufficient to uniquely identify the object.

UI Elements	Description
Keep the original description and continue to the next step	Keeps the original step containing the original object and continues to run the Maintenance Run Wizard on the remainder of the test or component. The Smart Identification page does not open for this object again during the run.
Apply this selection to all objects using Smart Identification in this run	Uses your radio button selection above for all objects in the test or component that need the Smart Identification mechanism to be identified.

Maintenance Mode Summary Page (Maintenance Run Wizard)

Relevant for: GUI tests and components

This wizard page enables you to view a summary of the Maintenance Mode run.



The Maintenance Run completed successfully.

1 object was added to the local **object repository**.

2 object **properties were** updated.

2 **steps were** modified.

0 **comments were** added to the test.

Wizard map	"Maintenance Run Wizard Workflow" on page 1087

See also	The wizard contains the following additional pages: • "Object Not Found Page (Maintenance Run Wizard)" on page 1092 • "Add Comment Page (Maintenance Run Wizard)" on page 1094 • "Change Object Property Values Page (Maintenance Run Wizard)" on page 1096 • "Update Step with Existing Object Page (Maintenance Run Wizard)" on page 1099 • "Add Object to Repository Page (Maintenance Run Wizard)" on page 1101 • "Smart Identification Page (Maintenance Run Wizard)" on page 1104

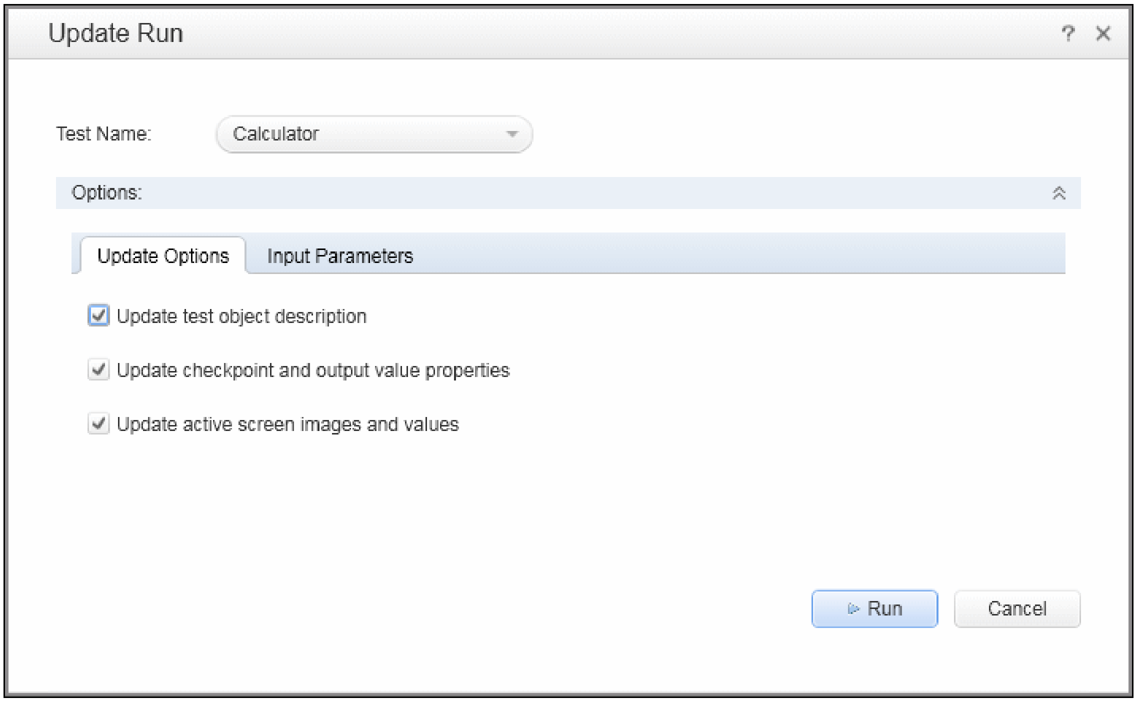
The Maintenance Mode Summary page displays the number of objects that were added to the local **object repository**, the number of object **properties** that were updated, the number of **steps** that were modified, and the number of **comments** that were added to the test.

Update Options Tab (Update Run Dialog Box)

Relevant for: GUI tests and components

The Update Options tab enables you to specify which aspects of your test or component you want to update, such as test object descriptions, expected checkpoint values, and/or, for tests and scripted components, Active Screen images and values. After you save the test or component, the results of the updated test or component are used for subsequent runs.

This image shows the Update Run dialog box for tests and scripted components. The **Update Active Screen images and values** option is not available for keyword components.



Update Run

Test Name: Calculator

Options:

Update Options | Input Parameters

☑ Update test object description

☑ Update checkpoint and output value properties

☑ Update active screen images and values

To access	1. Do one of the following: ▪ Ensure that a GUI test or component is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or component node. 2. Use one of the following: ▪ Select **Run > Update Run Mode**. ▪ Click the down arrow next to the **Run** button in the toolbar and select **Update Run Mode**.
Important information	See "Considerations for the Update Options Tab (Update Run Dialog Box)" on the next page
Relevant tasks	"How to Update Test Object Descriptions, Checkpoints, or Output Values, or Active Screen Captures" on page 1088
See also	• "Update Run Mode " on page 1081 • "Run Dialog Box: Input Parameters Tab (For GUI and API tests and components)" on page 655

User interface elements are described below:

UI Elements	Description
Update test object description	UFT updates the set of properties for each object class in your associated object repositories according to the properties currently defined in the "Object Identification Dialog Box" (described on page 1325). You can use this option to modify the set of properties used to identify an object of a certain type. For details, see "Update Run Mode " on page 1081.
Update checkpoint and output value properties	UFT updates the expected values of your checkpoints to reflect any changes that may have occurred in your application since you created the test and updates the list of items that can be retrieved in your output value steps. For details, see "When to Update Checkpoint Properties and Output Property Values" on page 1083.
Update active screen images and values (tests and scripted components only)	UFT updates images and property values in the Active Screen to reflect any changes that may have occurred in your application since you recorded the test or if the Active Screen does not appear as it should. For details, see "When to Update Active Screen Images and Values (tests and scripted components only)" on page 1084.

Considerations for the Update Options Tab (Update Run Dialog Box)

- When using the **Update test object descriptions**, if the property set you select in the "Object Identification Dialog Box" (described on page 1325) for an object class is not ideal for a particular object, the new object description may cause future runs to fail. Therefore, it is recommended that you save a copy of your object repositories (or check them into an ALM project with version control support, if applicable) before updating them, so that you can return to the previously saved version, if necessary.

- If checkpoint property values are parameterized or include regular expressions, they are not updated when using the **Update checkpoint properties and output property values** option.

- If you selected the **Save only selected area** check box when creating a bitmap checkpoint, the **Update Run Mode** option updates only the saved area of the bitmap; it does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint. For more details, see "Bitmap Checkpoints Overview" on page 1401.

Troubleshooting and Limitations - Maintenance Mode

Relevant for: GUI tests and components

This section describes troubleshooting and limitations for Maintenance Mode.

- When running in Maintenance Mode, if a step contains a programmatic description for an object that is not found in an application, it may take a while for Maintenance Mode to indicate there is a problem. If you use the option to point to the object, it may take a while for Maintenance Mode to reopen afterward.

- When the Maintenance Run Wizard cannot find an object, and the test object description for that object does not have any mandatory or assistive properties (it is identified only by its ordinal identifier, such as a Browser test object), then when you point to the object, the wizard is unable to fix the problem and displays a message that the object you pointed to has a test object description that is similar to the object that UFT could not identify.

  Workaround: Use the **Update from Application** option in the Object Repository window (for objects in the local respiratory) or in the Object Repository Manager (for objects in a shared object repository) to fix the test object description.

- When your Windows Display settings are set to use large fonts, the text in the Maintenance Run Wizard screens may be truncated.

- When the Maintenance Run Wizard cannot find an object in the application, and you point to a different object class to replace it, the Maintenance Run Wizard offers to add a step with that object and the object's default method. However, the wizard does not insert any method arguments for the step. If the step method has required arguments and you accept the step that the Maintenance Run Wizard proposes without modifying it, the step fails when you run it.

  Workaround: Enter valid method arguments for the step.

- 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.

- Maintenance Mode cannot fix problems in an object's properties, if the object is used in function library that is called by a step. If Maintenance Mode detects a problem in an object in a function library, a message is displayed indicating that the test is read-only and that Maintenance Mode is disabled.

Chapter 40: Recovery Scenarios for GUI Testing

Relevant for: GUI tests and components

This chapter includes:

Concepts

Recovery Scenarios Overview

Relevant for: GUI tests and components

Unexpected events, errors, and application crashes during a run session can disrupt your run session and distort results. This is a problem particularly when tests or components run unattended—the run pauses until you perform the operation needed to recover. To handle situations such as these, UFT enables you to create recovery scenarios and associate them with specific tests or application areas. Recovery scenarios activate specific recovery operations when trigger events occur.

The Recovery Scenario Manager provides a wizard that guides you through the process of defining a recovery scenario, which includes a definition of an unexpected event and the operations necessary to recover the run session. For example, you can instruct UFT to detect a **Printer out of paper** message and recover the run session by clicking the **OK** button to close the message and continue running.

A recovery scenario consists of the following:

- **Trigger Event.** The event that interrupts your run session. For example, a window that pops up on the screen, or a run error.

- **Recovery Operations.** The operations to perform to enable the run session to continue after the trigger event interrupts the session. For example, clicking an **OK** button in a pop-up window, or restarting Microsoft Windows.

- **Post-Recovery Test Run Option.** The instructions on how UFT should proceed after the recovery operations are performed, and from which step to continue, if at all. You may want to restart the run from the beginning, or skip a step entirely and continue with the next step.

After you create recovery scenarios, you associate them with selected tests or components (via the application area) so that the appropriate scenarios can run if a trigger event occurs. You can prioritize the scenarios and set the order in which to apply the scenarios during the run session. You can also choose to disable specific scenarios, or all scenarios, that are associated with a test or application area.

You can also define which recovery scenarios will be used as the default scenario for all new tests.

For tests: You can associate, remove, enable, disable, prioritize, and view the properties of the recovery scenarios associated with a GUI test in the Solution Explorer. For details, see "Solution Explorer Pane User Interface" on page 478.

For components: You define recovery scenarios for components in the application area. For details, see "Application Areas" on page 2095.

This section also includes:

When to Use Recovery Scenarios

Relevant for: GUI tests and components

Recovery scenarios are intended for use **only** with events that you cannot predict in advance, or for events that you cannot otherwise synchronize with a specific step in your test or component.

By default, recovery scenario operations are activated only after a step returns an error. This can potentially occur several steps after the step that originally caused the error. The alternative, checking for trigger events after every step, may slow performance. For this reason, it is best to handle predictable errors directly in your test or component.

If you can predict that a certain event may happen at a specific point in your test or component, it is highly recommended to handle that event directly within your test or component, rather than depending on a recovery scenario. To do this in a test, add steps such as If statements or optional steps. To do this in a component, use a user-defined function with conditional steps.

Handling an event directly within your test or component enables you to handle errors more specifically than recovery scenarios, which by nature are designed to handle a more generic set of unpredictable events. It also enables you to control the timing of the corrective operation with minimal resource usage and maximum performance.

Examples

- If you know that an Overwrite File message box may open when a **Save** button is clicked during a run session:

  You can handle this event with an If statement that clicks **OK** if the message box opens or by adding an optional step in the test to click **OK** in the message box. (For keyword components, you define this If statement in a user-defined function and make it available via the associated application area.)

- You can define a recovery scenario to handle printer errors. Then if a printer error occurs during a run session, the recovery scenario could instruct UFT to click the default button in the Printer Error message box.

  You would use a recovery scenario in this example because you cannot handle this type of error directly in your test or component. This is because you cannot know at what point the network will return the printer error. Even if you try to handle this event by adding an If statement in a user-defined function or in your test immediately after a step that sends a file to the printer, your test or component may progress several steps before the network returns the actual printer error.

Programmatically Controlling the Recovery Mechanism

Relevant for: GUI tests and components

You can use the Recovery object to control the recovery mechanism programmatically during the run session. For example, you can enable or disable the entire recovery mechanism or specific recovery scenarios for certain parts of a run session, retrieve status information about specific recovery scenarios, and explicitly activate the recovery mechanism at a certain point in the run session.

By default, UFT checks for recovery triggers when an error is returned during the run session. You can use the Recovery object's **Activate** method to force UFT to check for triggers after a specific step in the run session. For example, suppose you know that an object property checkpoint will fail if certain processes are open when the checkpoint is performed. You want to be sure that the pass or fail of the checkpoint is not affected by these open processes, which may indicate a different problem with your application.

However, a failed checkpoint does not result in a run error. So by default, the recovery mechanism would not be activated by the object state. You can define a recovery scenario that looks for and closes specified open processes when an object's properties have a certain state. This state shows the object's property values as they would be if the problematic processes were open. You can instruct UFT to activate the recovery mechanism if the checkpoint fails so that UFT can check for and close any problematic open processes and then perform the checkpoint again. This ensures that when the checkpoint is performed the second time it is not affected by the open processes.

For details on the Recovery object and its methods, see the *HP UFT Object Model Reference for GUI Testing*.

Tasks

How to Create and Manage Recovery Scenarios

Relevant for: GUI tests and components

This task describes how to perform different recovery scenario creation and management operations in the "Recovery Scenario Manager Dialog Box" (described on page "Recovery Scenario Manager Dialog Box").

This task includes the following:

- "Define a recovery scenario file in which to store the recovery scenarios" below
- "Create a new recovery scenario operation using the Recovery Scenario Wizard" below
- "Manage existing recovery scenarios" on the next page

Define a recovery scenario file in which to store the recovery scenarios

By default, the Recovery Scenario Manager dialog box opens with a new recovery file. You can use this new file, or you can open an existing recovery file, in one of the following ways:

- Click the arrow next to the **Open** button to select a recently-used recovery file from the list.
- Click the **Open** button and select an existing recovery file using the "Open/New <Document>/<Resource> Dialog Box" (described on page 156).

Create a new recovery scenario operation using the Recovery Scenario Wizard

1. In the "Recovery Scenario Manager Dialog Box" (described on page 1121), click the **New Scenario** button . The Recovery Scenario Wizard opens.
2. Follow the on-screen instructions. The wizard includes the following pages (pages that are in parentheses open according to the option selected in the previous page):
   - "Select Trigger Event Page " on page 1127
   - ("Specify Pop-up Window Conditions Page" on page 1130)
   - ("Select Object Page" on page 1131)
   - ("Set Object Properties and Values Page" on page 1133)
   - ("Select Test Run Error Page" on page 1136)
   - ("Select Processes Page" on page 1138)

  - "Recovery Operations Page" on page 1139
  - "Recovery Operation Page" on page 1142
  - ("Recovery Operation - Click Button or Press Key Page" on page 1143)
  - ("Recovery Operation - Close Processes Page" on page 1145)
  - ("Recovery Operation - Function Call Screen" on page 1146)
  - "Post-Recovery Test Run Options Page" on page 1148
  - "Name and Description Page" on page 1150
  - "Completing the Recovery Scenario Wizard Page" on page 1152

Manage existing recovery scenarios

- View properties for your recovery scenarios in the "Recovery Scenario Properties Dialog Box" (described on page 1124).
- Edit, save, and remove existing recovery scenarios in the Recovery Scenario Manager by using the toolbar buttons.

How to Manage Recovery Scenario Associations

Relevant for: GUI tests and components

This task describes how to perform different recovery scenario management and association operations using the Recovery pane in the Test Settings dialog box or an application area's Additional Settings pane. For a user interface description, see "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 613.

This task includes the following:

- "Associate a recovery scenario with your test in the Test Settings dialog box" on the next page
- "Associate a recovery scenario with your test in the Solution Explorer" on the next page
- "Associate a recovery scenario with your component or application area" on the next page
- "View read-only recovery scenario properties" on page 1118
- "Prioritize recovery scenarios" on page 1118
- "Remove recovery scenarios from your test or application area" on page 1118
- "Enable/disable specific recovery scenarios" on page 1119
- "Set default recovery scenario settings for all new tests" on page 1119

Associate a recovery scenario with your test in the Test Settings dialog box

1. In the "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" (described on page 613), click the **Add** button ➕.

2. In the "Add Recovery Scenario Dialog Box" (described on page 1120), click the **Browse** button and navigate to the recovery scenario file. A list of recovery scenario operations contained in this file opens in the "Add Recovery Scenario Dialog Box" (described on page 1120).

3. In the list of recovery scenarios, select a recovery scenario and click **Add Scenario**.

   The recovery scenario is displayed in the "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" (described on page 613) and is added under the "Recovery Scenarios Node" (described on page 488) found under your GUI test in the Solution Explorer.

Associate a recovery scenario with your test in the Solution Explorer

1. In the Solution Explorer, do one of the following:

   a. Right-click a GUI test node and select **Add > Associate Recovery Scenario.**

   b. Right-click the "Recovery Scenarios Node" and select **Associate Recovery Scenario**.

2. In the "Add Recovery Scenario Dialog Box" (described on page 1120), click the **Browse** button and navigate to the recovery scenario file. A list of recovery scenario operations contained in this file opens in the "Add Recovery Scenario Dialog Box".

3. In the list of recovery scenarios, select a recovery scenario and click **Add Scenario**.

   The recovery scenario is added under the "Recovery Scenarios Node" (described on page 488), found under your GUI test in the Solution Explorer.

Associate a recovery scenario with your component or application area

1. In Recovery pane of the Additional Settings pane of your application area, click the **Add** button ➕.

2. In the "Add Recovery Scenario Dialog Box" (described on page 1120), click the **Browse** button and navigate to the recovery scenario file. A list of all recovery scenario operations contained in this file opens in the Add Recovery Scenario dialog box.

3. In the list of recovery scenarios, select a recovery scenario and click **Add Scenario**.

   The recovery scenario is displayed in the Recovery pane in the Additional Settings pane of your application area and is associated with the component through its application area or is associated with the application area.

View read-only recovery scenario properties

In the Recovery Pane (Test/Component Settings / Application Area Additional Settings)

1. In the **Scenarios** box, select the recovery scenario whose properties you want to view.
2. Do one of the following:
   - Click the **Properties** button .
   - Double-click a scenario in the **Scenarios** box.

   The "Recovery Scenario Properties Dialog Box" (described on page 1124) opens, displaying read-only information regarding the settings for the selected scenario.

In the Solution Explorer:

Right-click the name of the recovery scenario whose properties you want to view and select **Recovery Scenario Properties.**

The "Recovery Scenario Properties Dialog Box" (described on page 1124) opens, displaying read-only information regarding the settings for the selected scenario.

Prioritize recovery scenarios

In the Recovery Pane (Test Settings / Application Area Additional Settings)

1. In the **Scenarios** box, select the scenario whose priority you want to change.
2. Click the **Up** or **Down** button ↑ ↓. The selected scenario's priority changes according to your selection.

In the Solution Explorer:

Right-click the scenario whose priority you want to change and select **Move Up** or **Move Down**. The selected scenario's priority changes according to your selection.

Remove recovery scenarios from your test or application area

In the Recovery Pane (Test Settings / Application Area Additional Settings)

1. In the **Scenarios** box, select the scenario you want to remove.
2. Click the **Remove** button ✕. The selected scenario is no longer associated with the test or application area.

In the Solution Explorer:

Right-click the scenario you want to remove and select **Remove Recovery Scenario from List.** The selected scenario is no longer associated with your test or application area and no longer appears under the test or application area node in the Solution Explorer.

Enable/disable specific recovery scenarios

Do one of the following:

- In the **Scenarios** box of the "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" (described on page 613), perform one of the following:
  - Select the check box to the left of one or more individual scenarios to enable them.
  - Clear the check box to the left of one or more individual scenarios to disable them.
- In the Solution Explorer, right-click the scenario you want to disable and select **Disable Recovery Scenario**.

Set default recovery scenario settings for all new tests

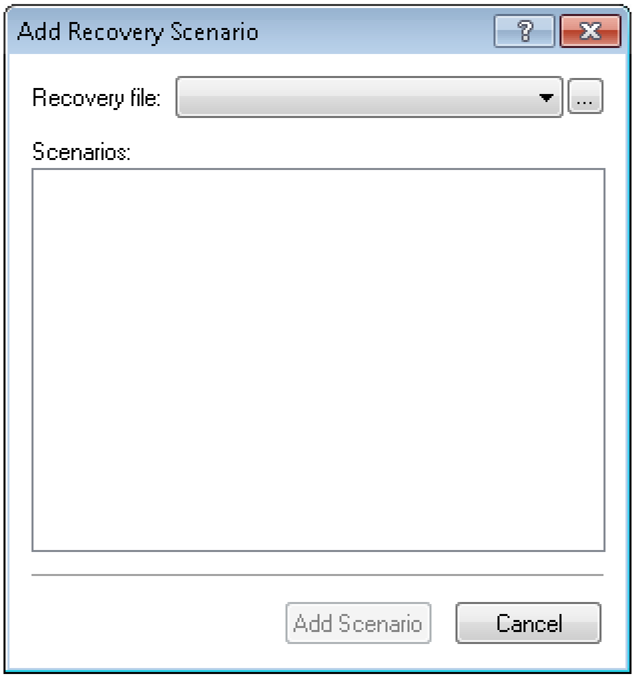
Click the **Set as Default** button in the Recovery pane of the Test Settings dialog box to set the current list of recovery scenarios to be the default scenarios for all new tests. Any future changes you make to the current recovery scenario list only affect the current test, and do not change the default list that you defined.

Reference

Add Recovery Scenario Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to associate existing scenarios with your test or your component (via its application area). This instructs UFT to perform the recovery scenarios during the run session.



To access	1. Do one of the following: ▪ Ensure that a GUI test or component is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or component node. 2. Use one of the following: ▪ **For tests:** Select **File > Settings > Recovery** node, and click the Add button + or right-click a GUI test node in the Solution Explorer and select **Add > Associate Recovery Scenario**. ▪ **For components:** In the **Additional Settings** pane of the application area, select the **Recovery** node, and click the **Add** button + .

Important information	If the recovery scenario you need is not available, you can create a new recovery scenario using the Recovery Scenario Manager (**Resources > Recovery Scenario Manager**). You can then associate it with your test or application area using the Add Recovery Scenario dialog box (described in this topic).
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1115
See also	• "Recovery Scenario Manager Dialog Box" below • "Recovery Scenario Wizard" on page 1126 • "Recovery Scenario Properties Dialog Box" on page 1124

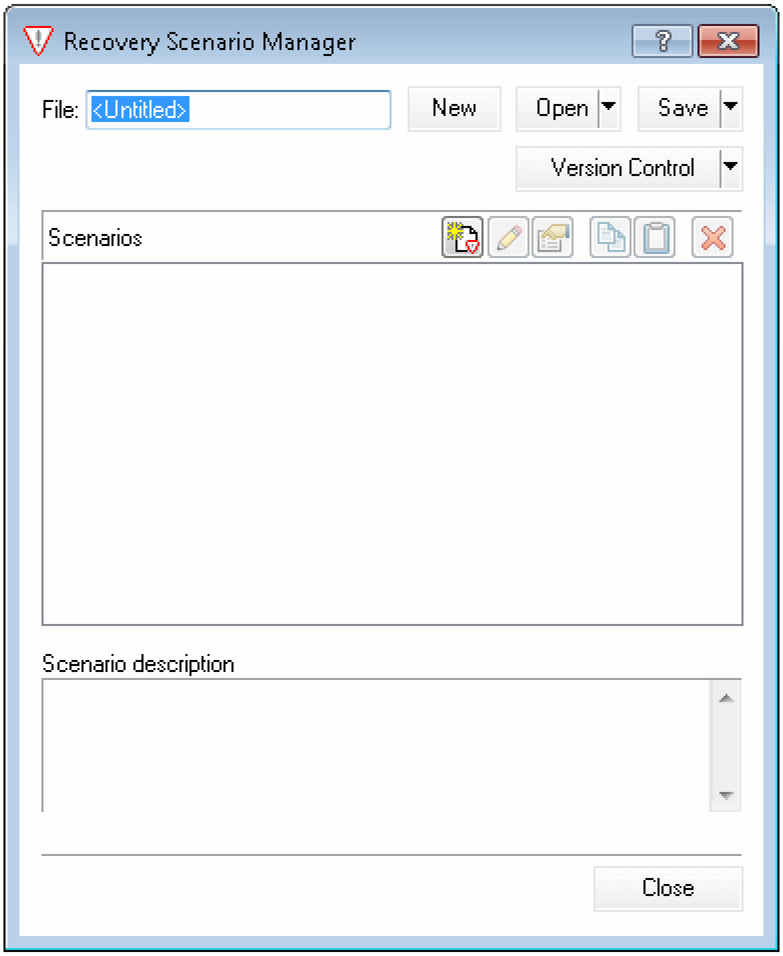
User interface elements are described below:

UI Elements	Description
Recovery file	The recovery file that contains the recovery scenario that you want to associate with the test or application area. You can click the **Browse** button to navigate to the recovery file you want to select.
Scenarios	The recovery scenarios in the recovery file selected in the Recovery file box.
Add Scenario	Associates the selected recovery scenario with the current test or application area.

Recovery Scenario Manager Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to create and edit recovery files, and create and manage the recovery scenarios stored in those files.



To access	1. Do one of the following: ▪ Ensure that a GUI test or component is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or component node. 2. Select **Resources** > **Recovery Scenario Manager**.
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1115
See also	• "Recovery Scenario Wizard" on page 1126 • "Recovery Scenario Properties Dialog Box" on page 1124

This dialog box contains the following key elements:

- "General User Interface Elements" on the next page
- "Scenarios Box Icons" on page 1124

General User Interface Elements

UI Elements	Description
File	The name of the selected recovery scenario file.
New	Creates a new recovery file.
Open	Opens an existing recovery file. You can also click the arrow to select a recovery file from the list of recently-used recovery files.
Save	Saves the current recovery file.
Version Control	Enables you to manage version control for your recovery scenarios. (Options are available only if UFT is connected to a version control-enabled ALM project.) For details, see "Managing Versions of Assets in ALM Overview" on page 795 and "Viewing and Comparing Versions of UFT Assets" on page 770.
Scenarios	The list of associated recovery scenario files. For details, see "Scenarios Box Icons" on the next page.
	New Scenario. Opens the Recovery Scenario Wizard, in which you define a new recovery scenario. For details, see "Recovery Scenario Wizard" on page 1126.
✎	**Edit.** Opens the Recovery Scenario Wizard for the selected recovery scenario, in which you can modify the recovery scenario settings. For details, see "Recovery Scenario Wizard" on page 1126.
	Properties. Displays summary properties for the selected recovery scenario in read-only format. For details, see "Recovery Scenario Properties Dialog Box" on the next page.
	Copy. Copies a recovery scenario from the open recovery file to the Clipboard. This enables you to paste a recovery scenario into another recovery file.
📋	**Paste.** Pastes a recovery scenario from the Clipboard into the open recovery file. **Note:** If a scenario with the same name already exists in the recovery scenario file, you can choose whether you want to replace it with the new scenario you have just copied.
✖	**Delete.** Deletes a recovery scenario. **Note:** If a deleted recovery scenario is associated with a test or component, UFT ignores it during the run session.
Scenario description	The description for the currently selected recovery scenario file.

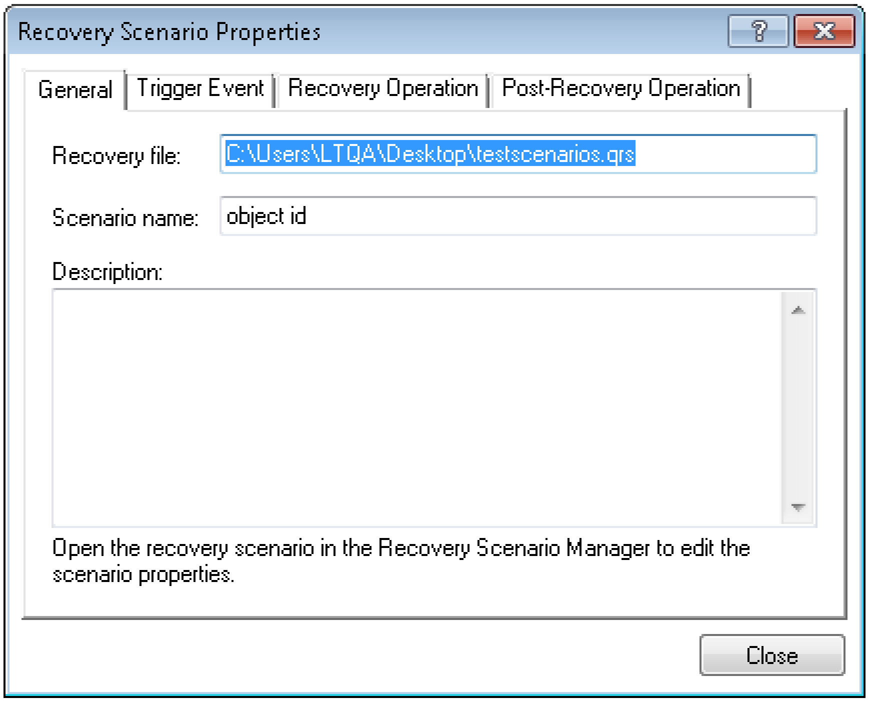
Scenarios Box Icons

Icon	Description
	Indicates that the recovery scenario is triggered when a window pops up in an open application during the run session.
	Indicates that the recovery scenario is triggered when the property values of an object in an application match specified values.
	Indicates that the recovery scenario is triggered when a step in the test or component does not run successfully.
	Indicates that the recovery scenario is triggered when an open application fails during the run session.

Recovery Scenario Properties Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to view read-only properties for any defined recovery scenario.



Recovery file: C:\Users\LTQA\Desktop\testscenarios.qrs

Scenario name: object id

Description:

Open the recovery scenario in the Recovery Scenario Manager to edit the scenario properties.

To access	- In the "Recovery Scenario Manager Dialog Box" (described on page 1121), select a scenario and click the **Properties** button . - In the "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)", double-click a scenario. - In the Solution Explorer, right-click a recovery scenario and select **Recovery Scenario Properties**.
Important information	Properties are displayed in read-only format. To modify any of the properties, use the **Edit** button in the Recovery Scenario Manager dialog box.
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1115
See also	- "Recovery Scenario Manager Dialog Box" on page 1121 - "Recovery Scenario Wizard" on the next page

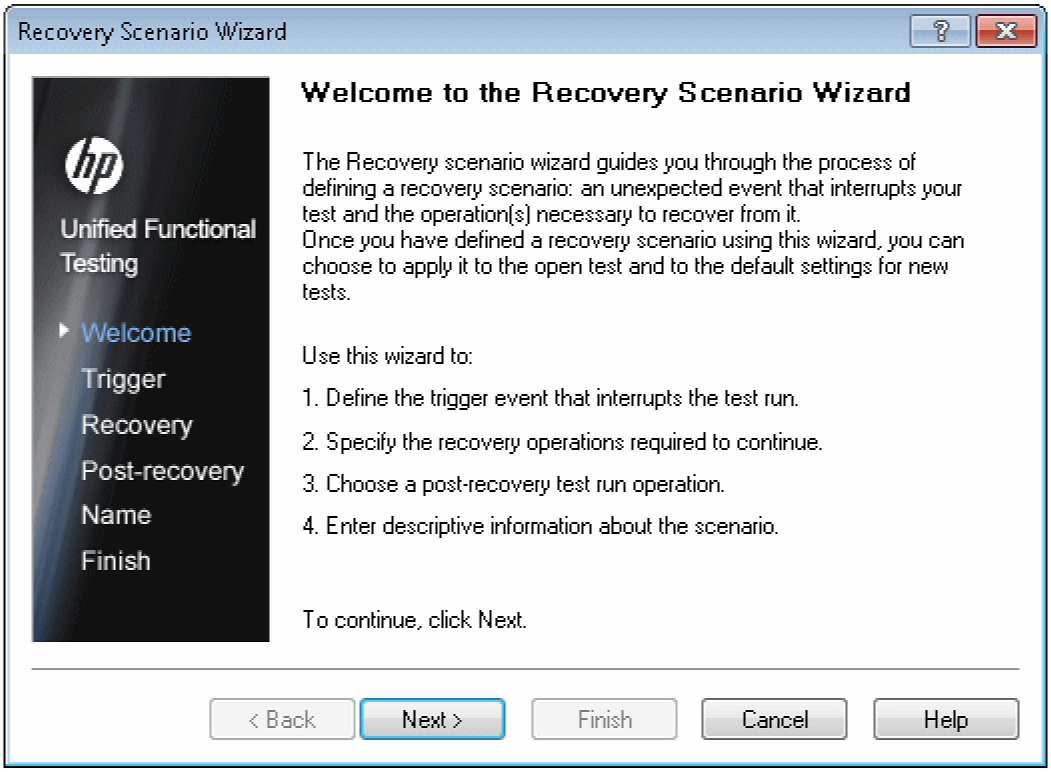
The Recovery Scenario Properties dialog box displays read-only information about the selected scenario in the following tabs:

UI Elements	Description
General	The name and description defined for the recovery scenario, plus the name and path of the recovery file in which the scenario is saved.
Trigger Event	The settings for the trigger event defined for the recovery scenario.
Recovery Operation	The recovery operations defined for the recovery scenario.
Post-Recovery Operation	The post-recovery operation defined for the recovery scenario.

Recovery Scenario Wizard

Relevant for: GUI tests and components

This wizard enables you to create a recovery scenario to associate with your test or application area.



Welcome to the Recovery Scenario Wizard

The Recovery scenario wizard guides you through the process of defining a recovery scenario: an unexpected event that interrupts your test and the operation(s) necessary to recover from it.
Once you have defined a recovery scenario using this wizard, you can choose to apply it to the open test and to the default settings for new tests.

Use this wizard to:

1. Define the trigger event that interrupts the test run.
2. Specify the recovery operations required to continue.
3. Choose a post-recovery test run operation.
4. Enter descriptive information about the scenario.

To continue, click Next.

To access	In the "Recovery Scenario Manager Dialog Box" (described on page 1121), click the **New Scenario** button .
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1115

Wizard map	This wizard contains: **Welcome** > "Select Trigger Event Page " (page 1127) > ("Specify Pop-up Window Conditions Page" (page 1130)) > ("Select Object Page" (page 1131)) > ("Set Object Properties and Values Page" (page 1133)) > ("Select Test Run Error Page" (page 1136)) > ("Select Processes Page" (page 1138)) > "Recovery Operations Page" (page 1139) > "Recovery Operation Page" (page 1142) > ("Recovery Operation - Click Button or Press Key Page" (page 1143)) > ("Recovery Operation - Close Processes Page" (page 1145)) > ("Recovery Operation - Function Call Screen" (page 1146)) > "Post-Recovery Test Run Options Page" (page 1148) > "Name and Description Page" (page 1150) > "Completing the Recovery Scenario Wizard Page" (page 1152) **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	"Recovery Scenarios Overview" on page 1112

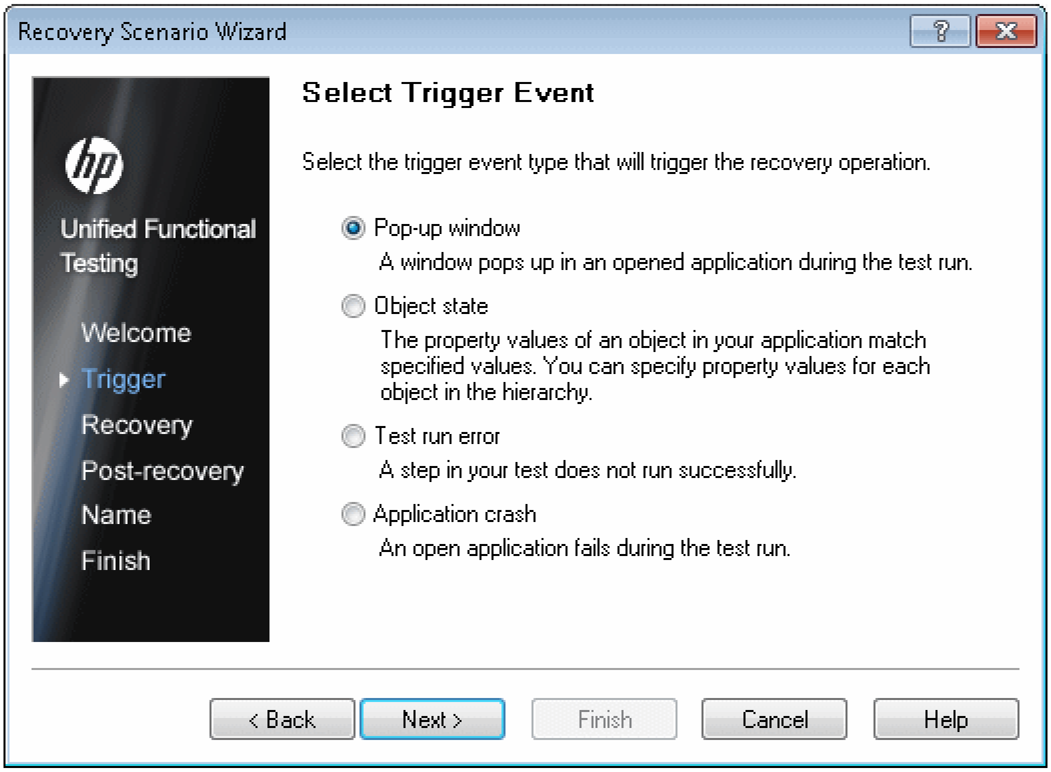
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<welcome area>	General information on the different options in the Recovery Scenario Wizard, and an overview of the stages involved in defining a recovery scenario.

Select Trigger Event Page

Relevant for: GUI tests and components

This page enables you to define the event type that triggers the recovery scenario, and the way in which UFT recognizes the event.



Select Trigger Event

Select the trigger event type that will trigger the recovery operation.

● Pop-up window
A window pops up in an opened application during the test run.

○ Object state
The property values of an object in your application match specified values. You can specify property values for each object in the hierarchy.

○ Test run error
A step in your test does not run successfully.

○ Application crash
An open application fails during the test run.

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > **Select Trigger Event Page** > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.
Important information	- The set of recovery operations is performed for each occurrence of the trigger event criteria. **Example:** If you define a specific object state, and two objects match this state, the set of recovery operations is performed two times, once for each object that matches the specified state. - The recovery mechanism does not handle triggers that occur in the last step of a test or component. If you need to recover from an unexpected event or error that may occur in the last step, you can do this by adding an extra step to the end of your test or component.

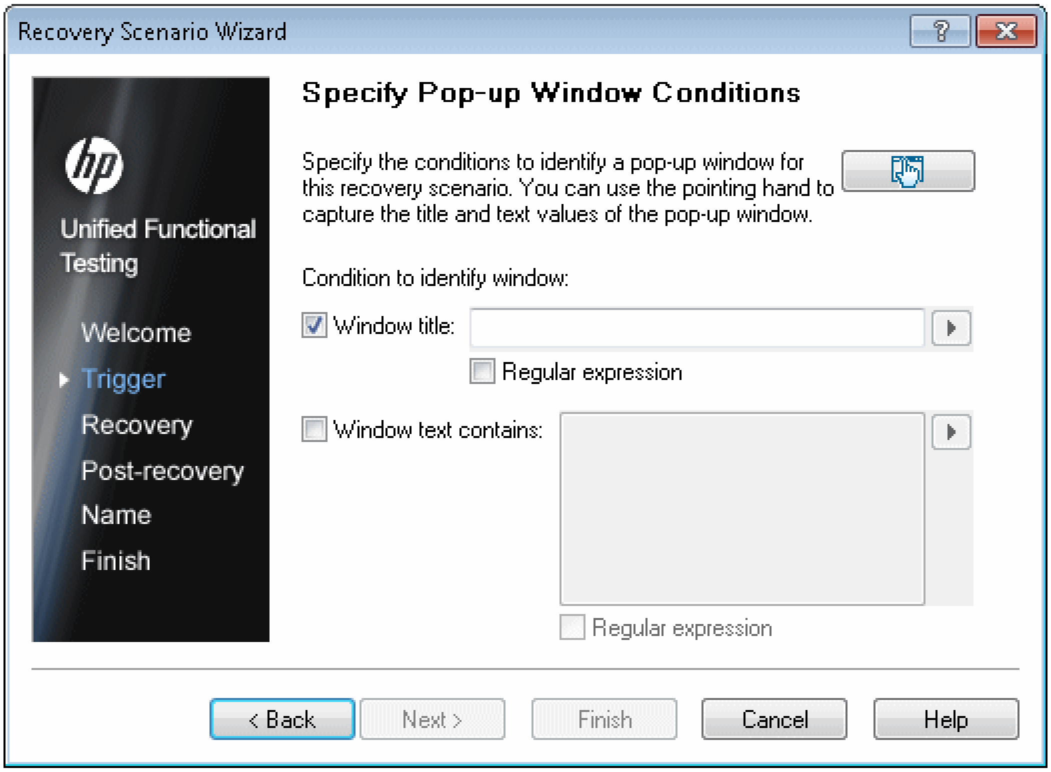
User interface elements are described below:

UI Elements	Description
Pop-up window	Detects a pop-up window and identifies it according to the window title and textual content. **Example:** A message box may open during a run session, indicating that the printer is out of paper. UFT can detect this window and activate a defined recovery scenario to continue the run session.
Object state	Detects a specific test object state and identifies it according to its property values and the property values of all its ancestors. Note that an object is identified only by its property values, and not by its class. **Example:** A specific button in a dialog box may be disabled when a specific process is open. UFT can detect the object property state of the button that occurs when this problematic process is open and activate a defined recovery scenario to close the process and continue the run session.
Test run error	Detects a run error and identifies it by a failed return value from a method. **Example:** UFT may not be able to identify a menu item specified in the method argument, due to the fact that the menu item is not available at a specific point during the run session. UFT can detect this run error and activate a defined recovery scenario to continue the run session.
Application crash	Detects an application crash and identifies it according to a predefined list of applications. **Example:** A secondary application may crash when a certain step is performed in the run session. You want to be sure that the run session does not fail because of this crash, which may indicate a different problem with your application. UFT can detect this application crash and activate a defined recovery scenario to continue the run session.

Specify Pop-up Window Conditions Page

Relevant for: GUI tests and components

This page enables you to specify how the pop-up window should be identified.



Recovery Scenario Wizard

Specify Pop-up Window Conditions

Specify the conditions to identify a pop-up window for this recovery scenario. You can use the pointing hand to capture the title and text values of the pop-up window.

Condition to identify window:

☑ Window title:

□ Regular expression

□ Window text contains:

□ Regular expression

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > (**Specify Pop-up Window Conditions Page**) > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

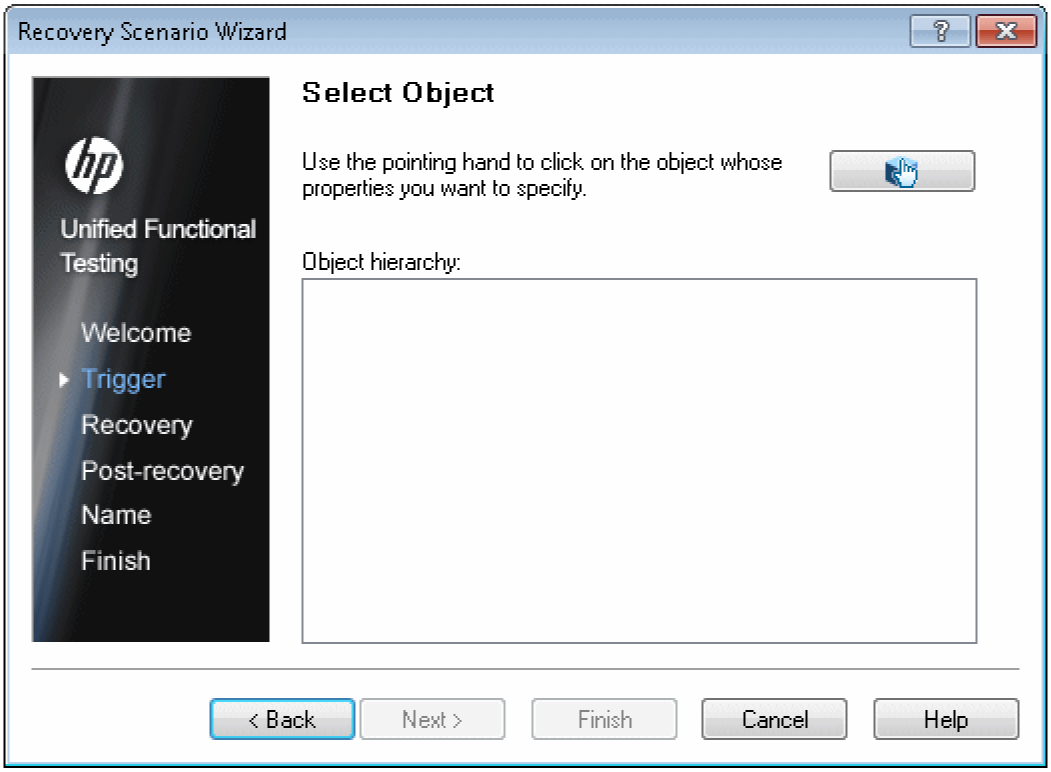
User interface elements are described below:

UI Elements	Description
☝	**Pointing hand**. Instructs UFT to identify only pop-up windows that match the object property values of the window you select. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.
Window title	Instructs UFT to identify any pop-up window that contains the relevant title.
Window text	Instructs UFT to identify any pop-up window that contains the relevant text.
Regular expression	Enables you to use regular expressions to identify the pop-up window. For details on regular expressions, see "Regular Expressions Overview" on page 318. **Note:** You can click the right arrow to display a list of regular expression characters that you can select, and to open the Regular Expression Evaluator, which enables you to test your regular expression. For details, see "Smart Regular Expression List" on page 361 and "Regular Expression Evaluator" on page 358.

Select Object Page

Relevant for: GUI tests and components

This page enables you to select the object whose properties you want to specify.



Select Object

Use the pointing hand to click on the object whose properties you want to specify.

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > (**Select Object Page**) > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

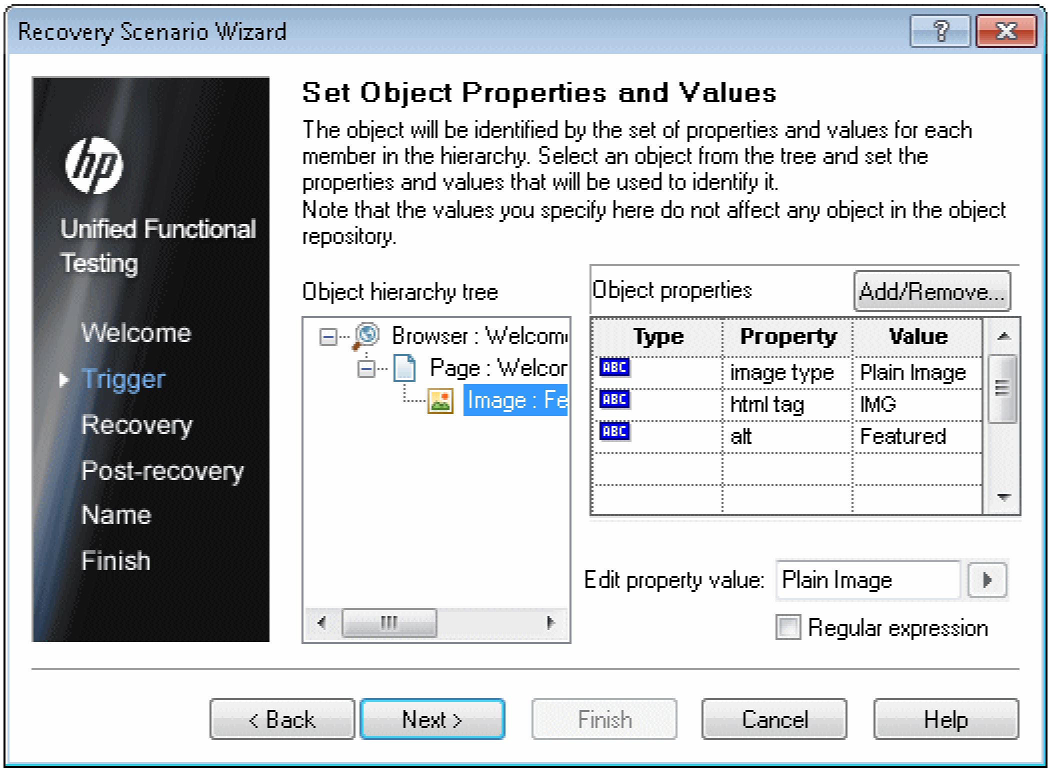
User interface elements are described below:

UI Elements	Description
👆	**Pointing hand.** Enables you to click the object whose properties you want to specify. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196. If the location you click is associated with more than one object, the "Object Selection Dialog Box " (described on page 1195) opens. Select the object whose properties you want to specify and click **OK**. The selected object and its parents are displayed in the Select Object screen. **Note:** The hierarchical object selection tree also enables you to select an object that UFT would not ordinarily learn (a non-parent object), such as a Web table.
Object hierarchy	The path to the selected object.

Set Object Properties and Values Page

Relevant for: GUI tests and components

This page enables you to specify the properties and values for the selected object to use in the recovery scenario.



Set Object Properties and Values

The object will be identified by the set of properties and values for each member in the hierarchy. Select an object from the tree and set the properties and values that will be used to identify it.
Note that the values you specify here do not affect any object in the object repository.

Object hierarchy tree

- Browser : Welcom
  - Page : Welcor
    - Image : Fe

Object properties

Type	Property	Value
ABC	image type	Plain Image
ABC	html tag	IMG
ABC	alt	Featured

Edit property value: Plain Image

□ Regular expression

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > (**Set Object Properties and Values Page**) > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

User interface elements are described below:

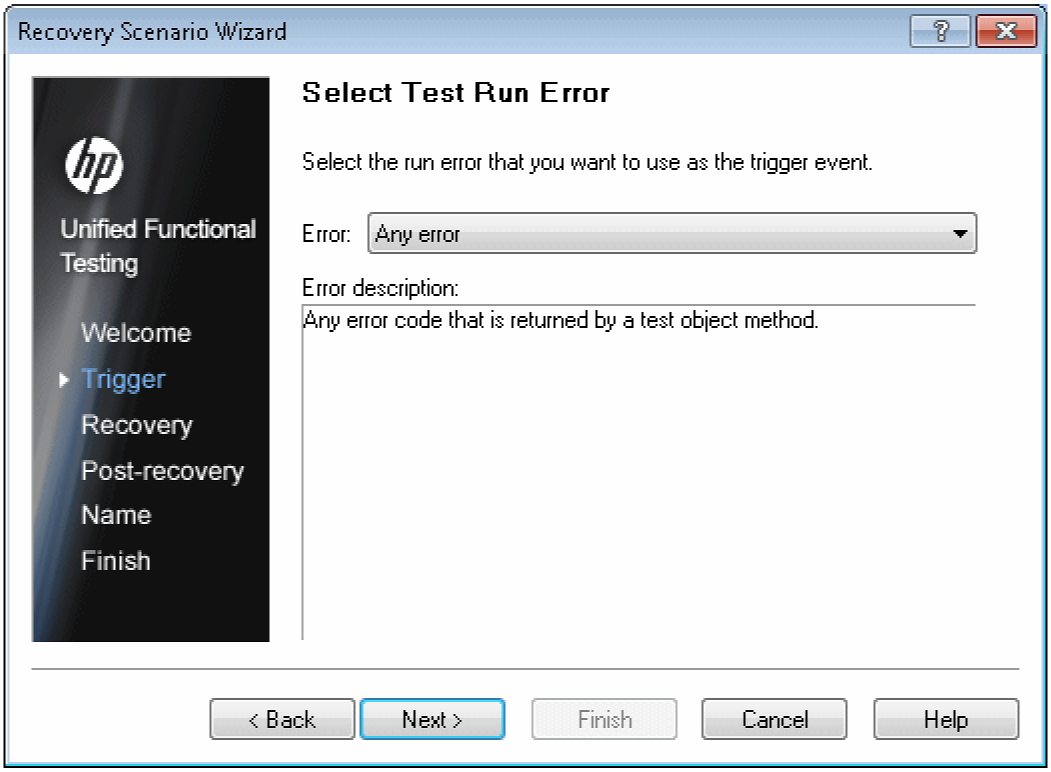
UI Elements	Description
Object hierarchy tree	The path to the selected object.
Object properties	The list of available properties for the selected object.

UI Elements	Description
Add/Remove..	Enables you to add or remove object properties from the list of property values to check. **Note:** An object is identified only by its property values, and not by its class.
Edit property value	Enables you to modify the property values used to identify the object.
Regular expression	Enables you to use regular expressions in the property value. For details on regular expressions, see "Regular Expressions Overview" on page 318. **Note:** You can click the right arrow to display a list of regular expression characters that you can select, and to open the Regular Expression Evaluator, which enables you to test your regular expression. For details, see "Smart Regular Expression List" on page 361 and "Regular Expression Evaluator" on page 358.

Select Test Run Error Page

Relevant for: GUI tests and components

This page enables you to select the type of error to use as a trigger event.



Select Test Run Error

Select the run error that you want to use as the trigger event.

Error: Any error

Error description:
Any error code that is returned by a test object method.

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > (**Select Test Run Error Page**) > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

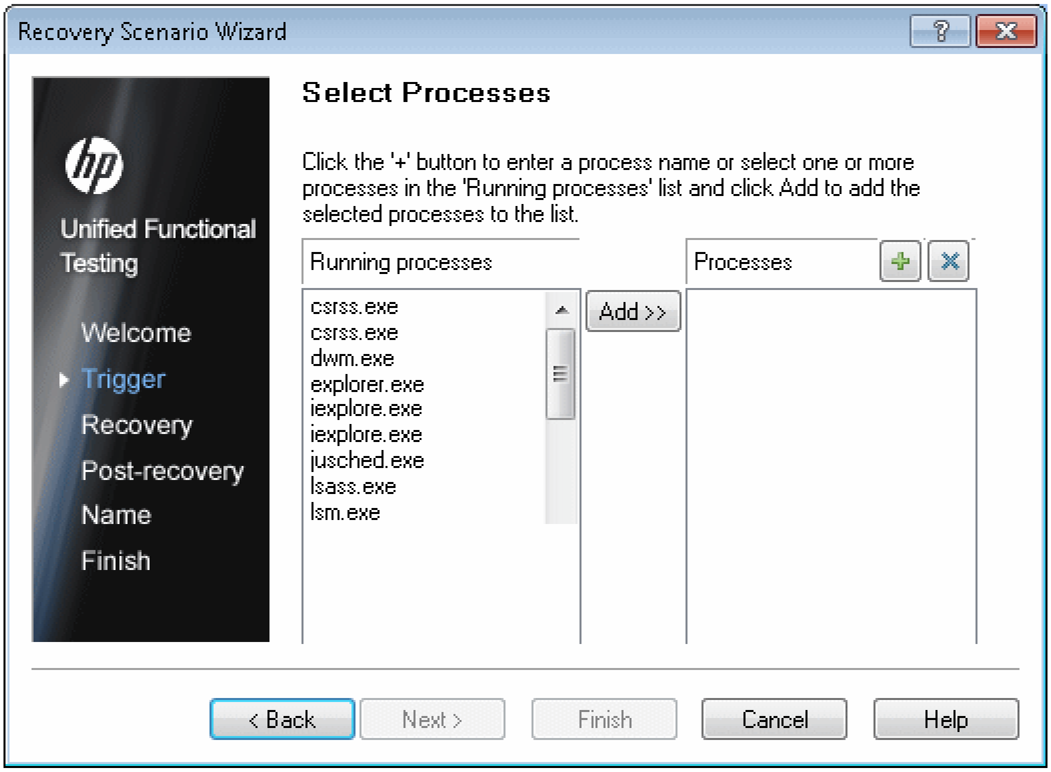
User interface elements are described below:

UI Elements	Description
Error	The list of error types to use as trigger events. The following option are available: • **Any error.** Any error code that is returned by a test object method. • **Item in list or menu is not unique.** Occurs when more than one item in the list, menu, or tree has the name specified in the method argument. • **Item in list or menu not found.** Occurs when UFT cannot identify the list, menu, or tree item specified in the method argument. This may be due to the fact that the item is not currently available or that its name has changed. • **More than one object responds to the physical description.** Occurs when more than one object in your application has the same property values as those specified in the test object description for the object specified in the step. • **Object is disabled.** Occurs when UFT cannot perform the step because the object specified in the step is currently disabled. • **Object not found.** Occurs when no object within the specified parent object matches the test object description for the object. • **Object not visible.** Occurs when UFT cannot perform the step because the object specified in the step is not currently visible on the screen.
Error description	The description of the selected error type.

Select Processes Page

Relevant for: GUI tests and components

This page enables you to select processes that will trigger a recovery scenario if they crash.



Select Processes

Click the '+' button to enter a process name or select one or more processes in the 'Running processes' list and click Add to add the selected processes to the list.

csrss.exe
csrss.exe
dwm.exe
explorer.exe
iexplore.exe
iexplore.exe
jusched.exe
lsass.exe
lsm.exe

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > (**Select Processes Page**) > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

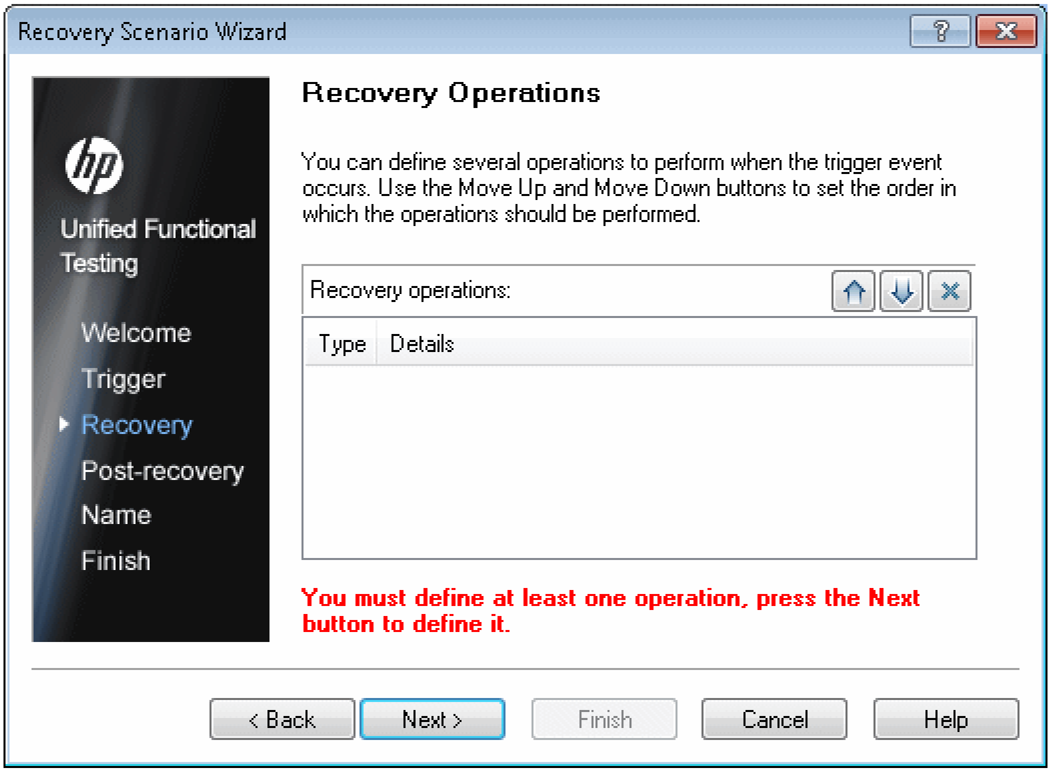
User interface elements are described below:

UI Elements	Description
Running processes	The list of all application processes that are currently running. To add a process from the **Running processes** list, double-click a process in the **Running processes** list or select it and click the **Add** button. You can select multiple processes using standard Windows multiple selection techniques (CTRL and SHIFT keys).
Processes	The list of the application processes that will trigger the recovery scenario if they crash. **Note:** • You can add application processes to the **Processes** list by typing them in the **Processes** list or by selecting them from the **Running processes** list. • You can modify the name of a process by selecting it in the **Processes** list and clicking the process name to edit it.
✚	**Add new process**. Enables you to add a process directly to the **Processes** list by entering the name of any process you want to add to the list.
✖	**Remove process**. Enables you to remove a process from the **Processes** list.

Recovery Operations Page

Relevant for: GUI tests and components

This page enables you to manage the collection of recovery operations in the recovery scenario.



Recovery Operations

You can define several operations to perform when the trigger event occurs. Use the Move Up and Move Down buttons to set the order in which the operations should be performed.

Recovery operations:

Type	Details

You must define at least one operation, press the Next button to define it.

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > **Recovery Operations Page** > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

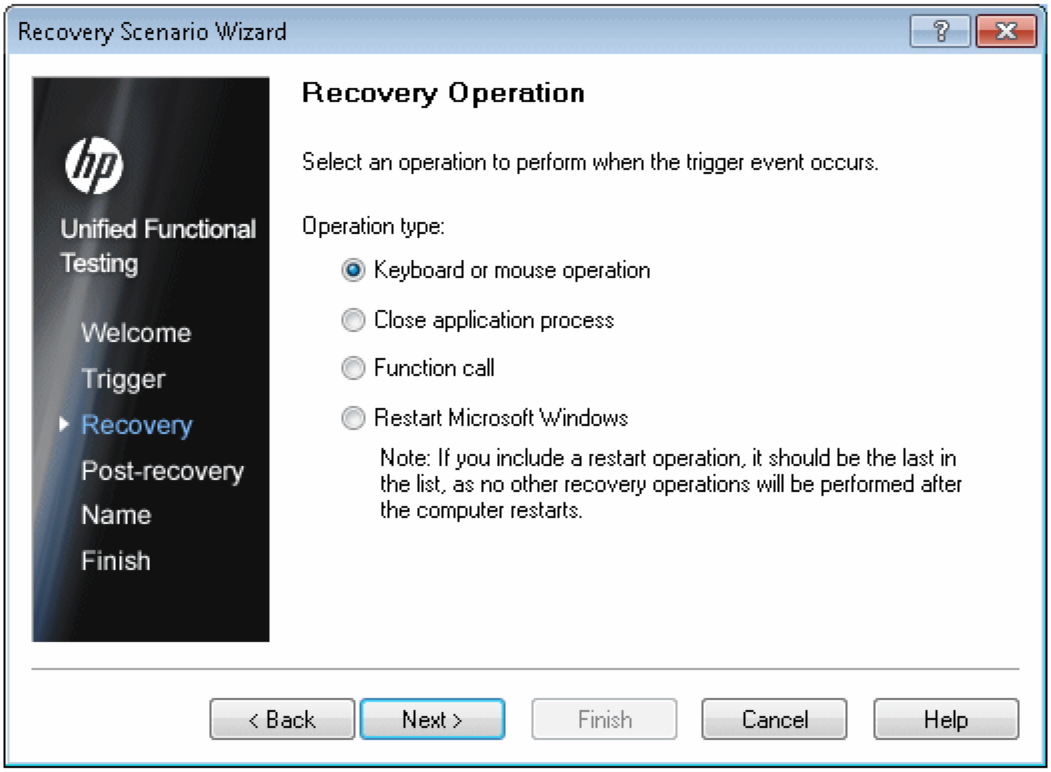
User interface elements are described below:

UI Elements	Description
Recovery operations	The list of selected recovery operations. To add a recovery scenario to the list, click **Next** to continue to the "Recovery Operation Page" (described on page 1142). **Note:** • If you define two or more recovery operations, you can select a recovery operation and use the **Move Up** or **Move Down** buttons to change the order in which to perform the recovery operations. • You can also select a recovery operation and click the **Remove** button to delete a recovery operation from the recovery scenario. • If you define a **Restart Microsoft Windows** recovery operation, it is always inserted as the last recovery operation, and you cannot change its position in the list.
Add another recovery operation	Displayed only after you have defined at least one recovery operation. The following options are available: • Select the check box and click **Next** to define another recovery operation. • Clear the check box and click **Next** to continue to the "Post-Recovery Test Run Options Page" (described on page 1148).

Recovery Operation Page

Relevant for: GUI tests and components

This page enables you to specify the operations UFT performs after it detects the trigger event.



Recovery Operation

Select an operation to perform when the trigger event occurs.

Operation type:

● Keyboard or mouse operation
○ Close application process
○ Function call
○ Restart Microsoft Windows

Note: If you include a restart operation, it should be the last in the list, as no other recovery operations will be performed after the computer restarts.

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > **Recovery Operation Page** > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

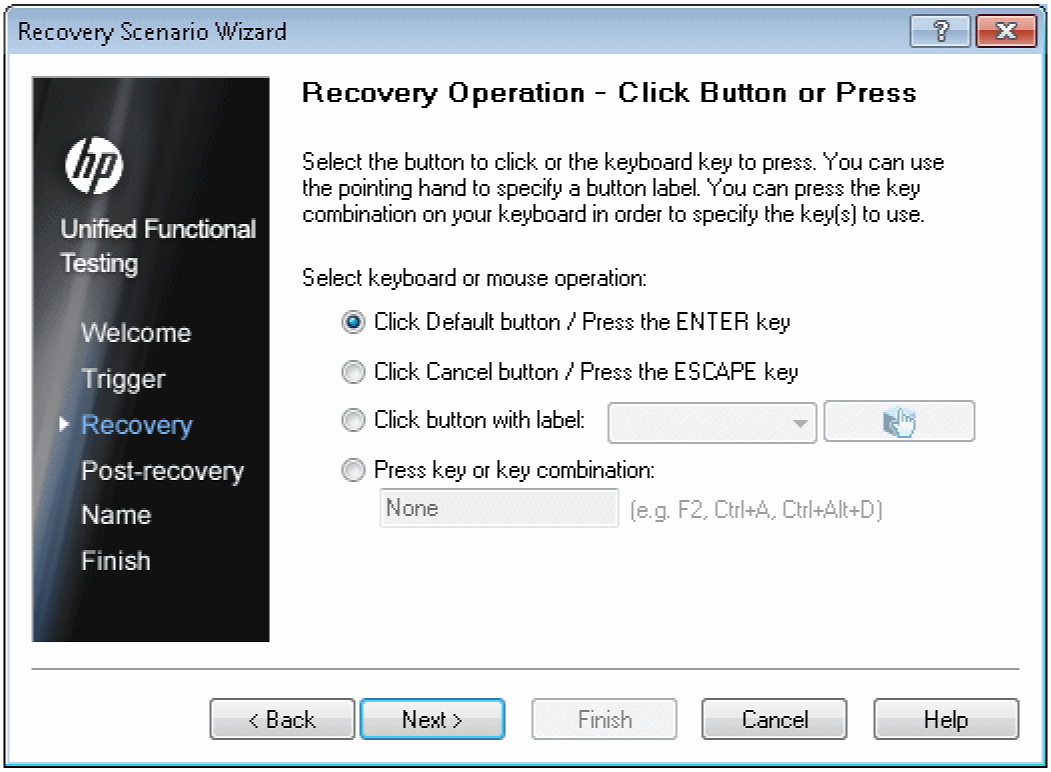
User interface elements are described below:

UI Elements	Description
Keyboard or mouse operation	Simulates a click on a button in a window or a press of a keyboard key.
Close application process	Closes specified processes.
Function call	Calls a VBScript function.
Restart Microsoft Windows	Restarts Microsoft Windows. **Note:** If you use the this recovery operation, you must ensure that any test or application area (and component) associated with this recovery scenario is saved before you run it. You must also configure the computer on which the test or component is run to automatically log in on restart.

Recovery Operation - Click Button or Press Key Page

Relevant for: GUI tests and components

This page enables you to specify the keyboard or mouse operation that you want UFT to perform when it detects the trigger event:



Recovery Scenario Wizard

Recovery Operation - Click Button or Press

Select the button to click or the keyboard key to press. You can use the pointing hand to specify a button label. You can press the key combination on your keyboard in order to specify the key(s) to use.

Select keyboard or mouse operation:

- ● Click Default button / Press the ENTER key
- ○ Click Cancel button / Press the ESCAPE key
- ○ Click button with label:
- ○ Press key or key combination:

None (e.g. F2, Ctrl+A, Ctrl+Alt+D)

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > (**Recovery Operation - Click Button or Press Key Page**) > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

User interface elements are described below:

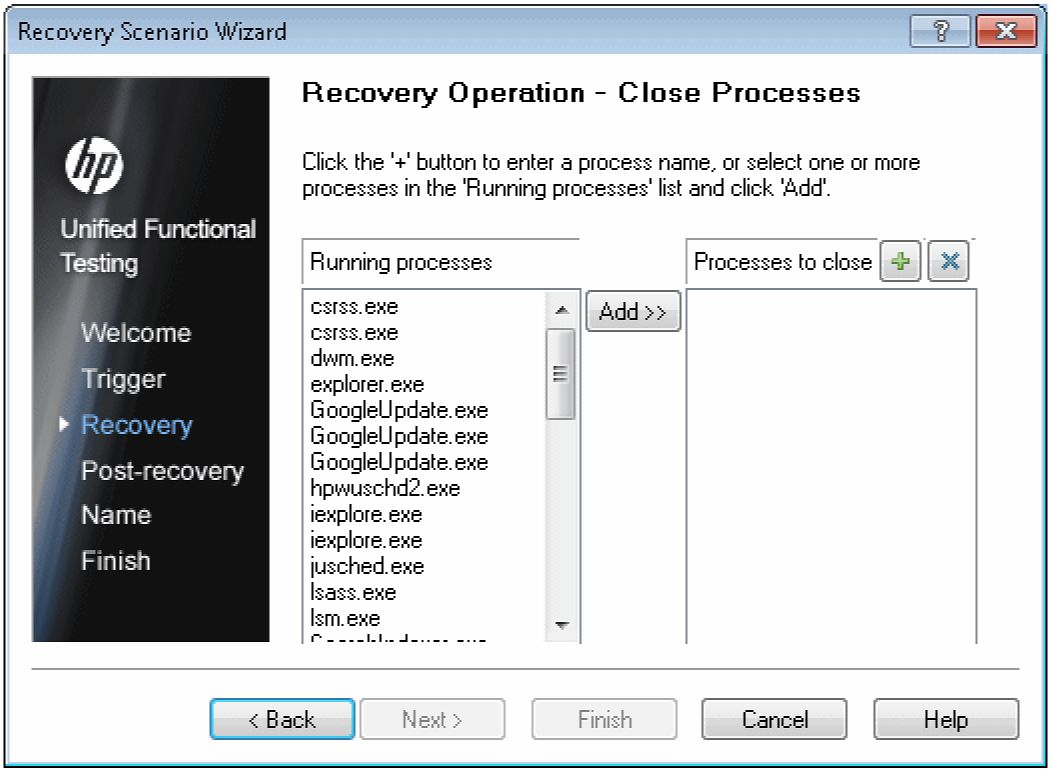
UI Elements	Description
Click Default button / Press the ENTER key	Clicks the default button or presses the Enter key in the displayed window when the trigger occurs.

UI Elements	Description
Click Cancel button / Press the ESCAPE key	Clicks the Cancel button or presses the Escape key in the displayed window when the trigger occurs.
Click button with label	Clicks the button with the specified label in the displayed window when the trigger occurs. If you select this option, click the pointing hand and then click anywhere in the trigger window. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196. All button labels in the selected window are displayed in the list box. Select the required button from the list.
Press key or key combination	Presses the specified keyboard key or key combination in the displayed window when the trigger occurs. If you select this option, click in the edit box and then press the key or key combination on your keyboard that you want to specify.

Recovery Operation - Close Processes Page

Relevant for: GUI tests and components

This page enables you to select processes to close in the recovery operation.



Recovery Scenario Wizard

Recovery Operation - Close Processes

Click the '+' button to enter a process name, or select one or more processes in the 'Running processes' list and click 'Add'.

Running processes

csrss.exe
csrss.exe
dwm.exe
explorer.exe
GoogleUpdate.exe
GoogleUpdate.exe
GoogleUpdate.exe
hpwuschd2.exe
iexplore.exe
iexplore.exe
jusched.exe
lsass.exe
lsm.exe

Processes to close

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > (**Recovery Operation - Close Processes Screen**) > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

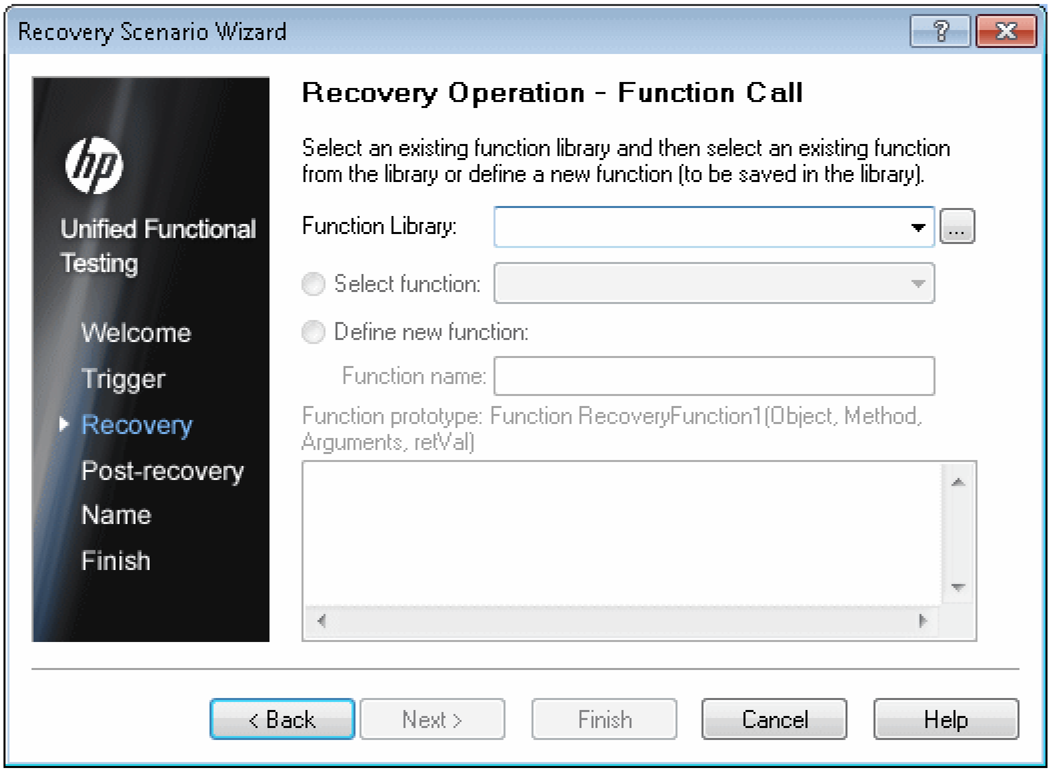
User interface elements are described below:

UI Elements	Description
Running processes	The list of all application processes that are currently running. To add a process from the **Running processes** list, double-click a process in the **Running processes** list or select it and click the **Add** button. You can select multiple processes using standard Windows multiple selection techniques (CTRL and SHIFT keys).
Processes to close	The list of the application processes that will be closed when the trigger is activated. **Note:** You can modify the name of a process by selecting it in the **Processes to close** list and clicking the process name to edit it.
✚	**Add new process**. Enables you to add a process directly to the **Processes** list by entering the name of any process you want to add to the list.
✖	**Remove process**. Enables you to remove a process from the **Processes** list.

Recovery Operation - Function Call Screen

Relevant for: GUI tests and components

This page enables you to select a function to call in the recovery operation.



Recovery Operation - Function Call

Select an existing function library and then select an existing function from the library or define a new function (to be saved in the library).

Function Library:

○ Select function:

○ Define new function:

Function name:

Function prototype: Function RecoveryFunction1(Object, Method, Arguments, retVal)

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > (**Recovery Operation - Function Call Page**) > "Post-Recovery Test Run Options Page" > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.
Important information	• If more than one scenario uses a function with the same name from different function libraries, the recovery process may fail. In this case, information regarding the recovery failure is displayed during the run session. • **For tests:**UFT automatically associates the function library you select with your test. Therefore, you do not need to associate the function library with your test in the Resources pane of the Test Settings dialog box. • **For components:** The function library must be stored in the ALM project.

User Interface Elements

UI Elements	Description
Function Library	Displays the selected function library, and enables you to select a recently specified function library in the box. Alternatively, click the **Browse** button to navigate to an existing function library.
Select function	Choose an existing function from the function library you selected. Only functions that match the prototype syntax for the trigger type selected in the "Select Trigger Event Page " on page 1127 are displayed.
Define new function	Create a new function by specifying a unique name for it, and defining the function in the **Function Name** box according to the displayed function prototype. The new function is added to the function library you selected.

Function Prototypes by Trigger Type

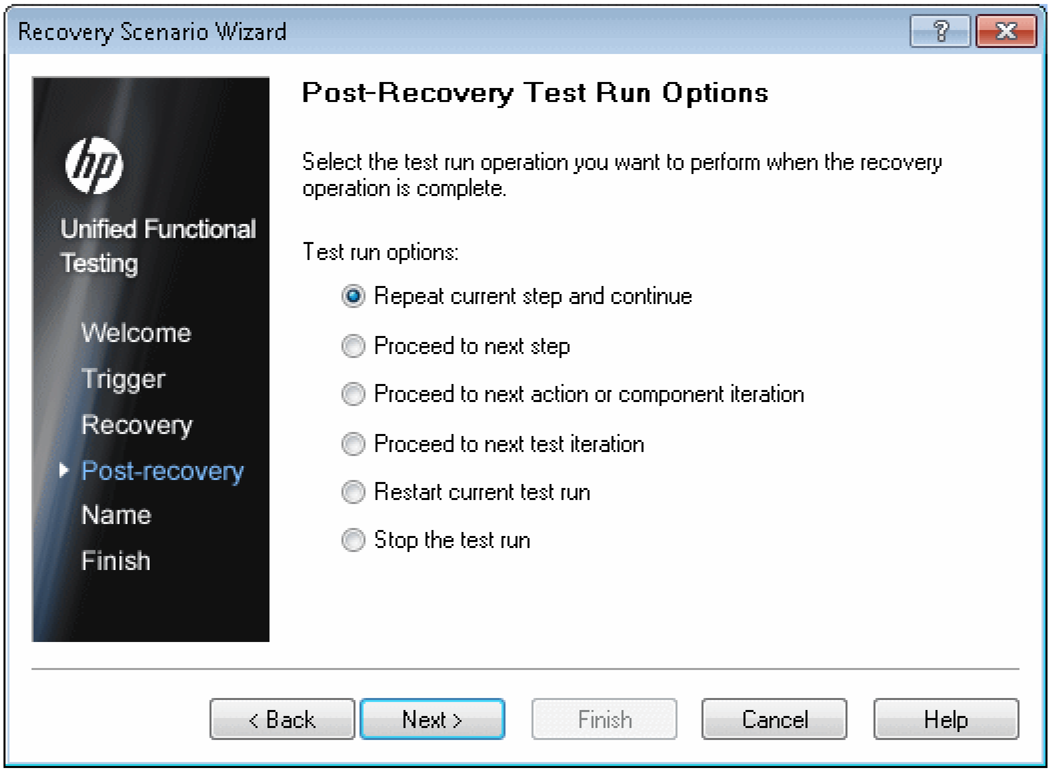
Following is the prototype for each trigger type:

```
Test run error trigger
OnRunStep
(
[in]  Object as Object: The object of the current step.
[in]  Method as String: The method of the current step.
[in]   Arguments as Array: The actual method's arguments.
[in]   Result as Integer: The actual method's result.
)
Pop-up window and Object state triggers
OnObject
(
[in]  Object as Object: The detected object.
)Application crash trigger
OnProcess
(
[in]  ProcessName as String: The detected process's Name.
[in]  ProcessId as Integer: The detected process' ID.
)
```

Post-Recovery Test Run Options Page

Relevant for: GUI tests and components

This page enables you to define post-recovery test run options, which specify how to continue the run session after UFT identifies the event and performs all of the specified recovery operations.



Post-Recovery Test Run Options

Select the test run operation you want to perform when the recovery operation is complete.

Test run options:

- ● Repeat current step and continue
- ○ Proceed to next step
- ○ Proceed to next action or component iteration
- ○ Proceed to next test iteration
- ○ Restart current test run
- ○ Stop the test run

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > **Post-Recovery Test Run Options Page** > "Name and Description Page" > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.
Important information	If you chose **Restart Microsoft Windows** as a recovery operation, you can choose from only the last two test run options.

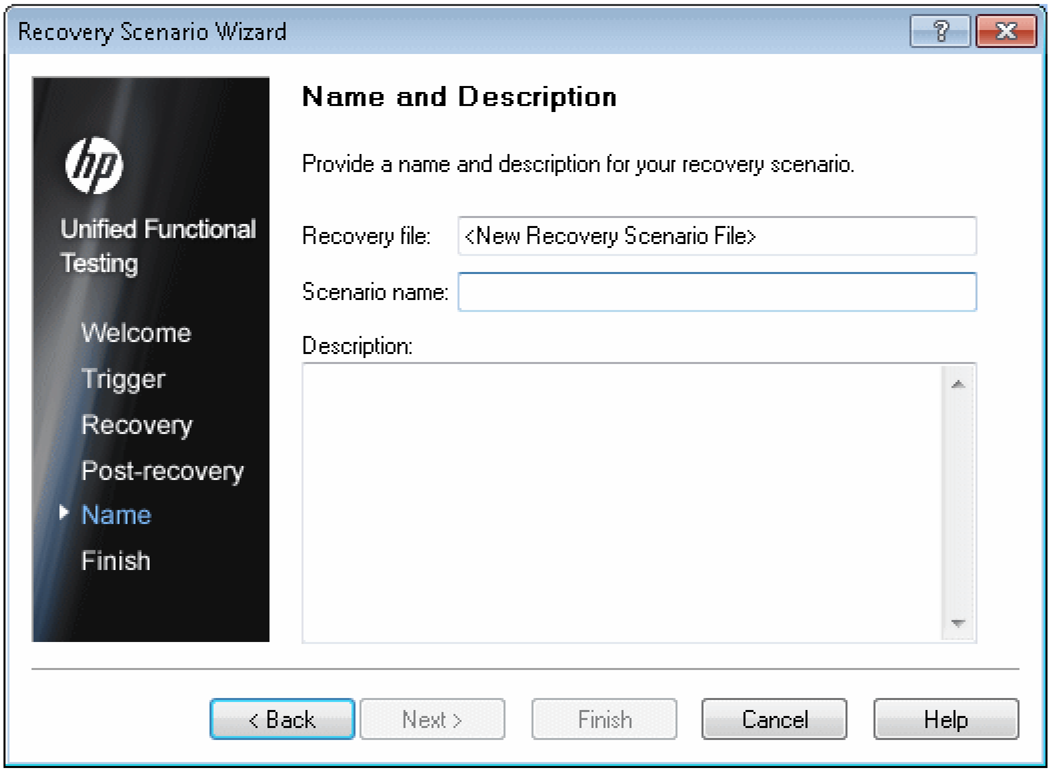
User interface elements are described below:

UI Elements	Description
Repeat current step and continue	The current step is the step that UFT was running when the recovery scenario was triggered. If you are using the **On error** activation option for recovery scenarios, the step that returns the error is often one or more steps later than the step that caused the trigger event to occur. Therefore, in most cases, repeating the current step does not cause the trigger event to occur again.
Proceed to next step	Skips the step that UFT was running when the recovery scenario was triggered. Keep in mind that skipping a step that performs operations on your application may cause subsequent steps to fail.
Proceed to next action or component iteration	Stops performing steps in the current action or component iteration and begins the next iteration from the beginning (or from the next action or component if no additional iterations of the current action or component are required).
Proceed to next test iteration	Stops performing steps in the current action or component and begins the next GUI test or business process test iteration from the beginning (or stops the run session if no additional iterations are required).
Restart current test run	Stops performing steps and re-runs the GUI test from the beginning. (Not applicable to business process tests.)
Stop the test run	Stops running the test or component.

Name and Description Page

Relevant for: GUI tests and components

This page enables you to specify a name by which to identify your recovery scenario. You can also add descriptive information regarding the scenario.



Name and Description

Provide a name and description for your recovery scenario.

Recovery file: <New Recovery Scenario File>

Scenario name:

Description:

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > **Name and Description Page** > "Completing the Recovery Scenario Wizard Page" **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

User interface elements are described below:

UI Elements	Description
Recovery file	The name of the recovery file.
Scenario name	The name of the scenario operation.
Description	The textual description of the scenario operation.

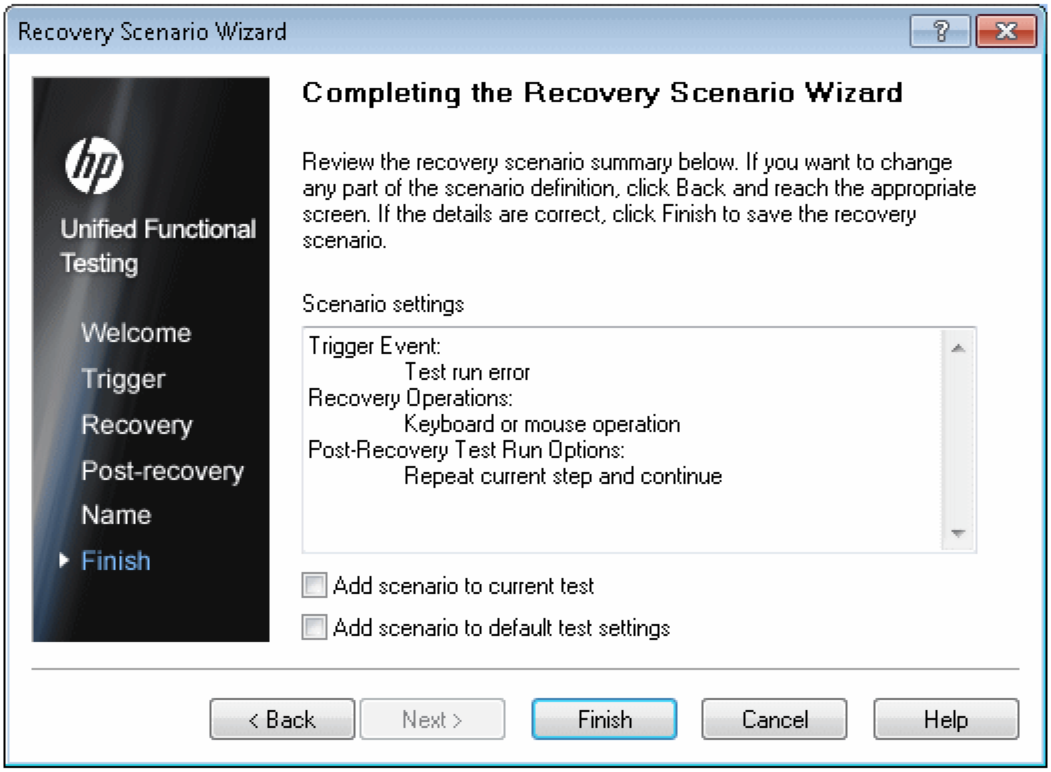
Completing the Recovery Scenario Wizard Page

Relevant for: GUI tests and components

This page enables you to review a summary of the scenario settings you defined.

For tests: You can also specify whether to automatically associate the recovery scenario with the current test and/or to add it to the default settings for all new tests.

The following image illustrates an example of the Recovery Scenario Wizard summary page. (The check boxes shown are available only if a test was in focus when you started the wizard.)



Completing the Recovery Scenario Wizard

Review the recovery scenario summary below. If you want to change any part of the scenario definition, click Back and reach the appropriate screen. If the details are correct, click Finish to save the recovery scenario.

Scenario settings

Trigger Event:
Test run error
Recovery Operations:
Keyboard or mouse operation
Post-Recovery Test Run Options:
Repeat current step and continue

□ Add scenario to current test

□ Add scenario to default test settings

Wizard map	The "Recovery Scenario Wizard" contains: Welcome > "Select Trigger Event Page " > ("Specify Pop-up Window Conditions Page") > ("Select Object Page") > ("Set Object Properties and Values Page") > ("Select Test Run Error Page") > ("Select Processes Page") > "Recovery Operations Page" > "Recovery Operation Page" > ("Recovery Operation - Click Button or Press Key Page") > ("Recovery Operation - Close Processes Page") > ("Recovery Operation - Function Call Screen") > "Post-Recovery Test Run Options Page" > "Name and Description Page" > **Completing the Recovery Scenario Wizard Screen** **Note:** Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

User interface elements are described below:

UI Elements	Description
Scenario settings	The settings of the defined recovery operation. **For components:** You associate a recovery scenario for a component with the component's application area. You can also define the default recovery scenarios for all new components associated with a specific application area. For details, see "Application Areas" on page 2095.
Add scenario to current test (tests only)	Associates this recovery scenario with the current test. When you click **Finish**, UFT adds the recovery scenario to the **Scenarios** list in the Recovery pane of the Test Settings dialog box.
Add scenario to default test settings (tests only)	Makes this recovery scenario a default scenario for all new tests. The next time you create a test, this scenario will be listed in the **Scenarios** list in the Recovery pane of the Test Settings dialog box. **Note:** You can remove scenarios from the default scenarios list. For details, see "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 613.

Troubleshooting and Limitations - Recovery Scenarios

Relevant for: GUI tests and components

This section describes troubleshooting and limitations for recovery scenarios.

- If you specify multiple function libraries from different locations with the same name in the same recovery scenario, only the first function library is used.

  Workaround: Rename the function libraries so that each function library has a unique name.

Chapter 41: UFT Automation Scripts

Relevant for: GUI tests and components

This chapter includes:

Concepts

UFT Automation Object Model Overview

Relevant for: GUI tests and components

You can use the UFT automation object model to write scripts that automate your UFT operations. The UFT automation object model provides objects, methods, and properties that enable you to control UFT from another application.

Using the objects, methods, and properties exposed by the UFT automation object model, you can write scripts that configure UFT options and run tests or components instead of performing these operations manually using the UFT interface.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple tests or components, or for quickly configuring UFT according to your needs for a particular environment or application.

What is Automation?

Automation is a Microsoft technology that makes it possible to access software objects inside one application from other applications. These objects can be created and manipulated using a scripting or programming language such as VBScript or VC++. Automation enables you to control the functionality of an application programmatically.

What is the UFT Automation Object Model?

An **object model** is a structural representation of software objects (classes) that comprise the implementation of a system or application. An object model defines a set of classes and interfaces, together with their properties, methods and events, and their relationships.

The UFT **automation object model** is a set of objects, methods, and properties that enable you to control essentially all of the configuration and run functionality provided via the UFT interface. Although a one-on-one comparison cannot always be made, most dialog boxes in UFT have a corresponding automation object, most options in dialog boxes can be set and/or retrieved using the corresponding object property, and most menu commands and other operations have corresponding automation methods.

You can use the objects, methods, and properties exposed by the UFT automation object model, along with standard programming elements such as loops and conditional statements to design your script.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple tests or components, or for quickly configuring UFT according to your needs for a particular environment or application.

Example:

You can create and run an automation script from Microsoft Visual Basic that does the following:

- Loads the required add-ins for a test or component
- Starts UFT in visible mode
- Opens the test or component
- Configures settings that correspond to those in the:
  - Options dialog box
  - Test Settings or Business Component Settings dialog box
  - Record and Run Settings dialog box (tests only)
- Runs the test or component
- Saves the test or component

You can then add a simple loop to your script so that your single script can perform the operations described above for multiple tests or components.

You can also create an initialization script that opens UFT with specific configuration settings. You can then instruct all of your testers to open UFT using this automation script to ensure that all of your testers are always working with the same configuration.

When to Use UFT Automation Scripts

Relevant for: GUI tests and components

Creating a useful UFT automation script requires planning, design time, and testing. You must always weigh the initial investment with the time and human-resource savings you gain from automating potentially long or tedious tasks.

Any UFT operation that you must perform many times in a row or must perform on a regular basis is a good candidate for a UFT automation script.

The following are just a few examples of useful UFT automation scripts:

- **Initialization scripts.** You can write a script that automatically starts UFT and configures the options and the settings required for testing a specific environment.
- **Maintaining your tests and components.** You can write a script that iterates over your collection of tests and components to accomplish a certain goal. For example:
  - **Updating values.** You can write a script that opens each test or component with the proper add-ins, runs it in update run mode against an updated application, and saves it when you want to update the values in all of your tests or components to match the updated values in your application.

  - **Applying new options to existing tests and components.** When you upgrade toUFT, you may find that it offers certain options that you want to apply to your existing tests and components. You can write a script that opens each existing test or component, sets values for the new options, then saves and closes it.
  - **Modifying Actions and Action Parameters (tests only).** You can retrieve the entire contents of an action script, and add a required step, such as a call to a new action. You can also retrieve the set of action parameters for an action and add, remove, or modify the values of action parameters.
- **Calling UFT from other applications.** You can design your own applications with options or controls that run UFT automation scripts. For example, you could create a Web form or simple Windows interface from which a product manager could schedule UFT runs, even if the manager is not familiar with UFT.

Application Object

Relevant for: GUI tests and components

Like most automation object models, the root object of the UFT automation object model is the **Application** object.

The **Application** object represents the application level of UFT. You can use this object to return other elements of UFT such as the:

- **Test** object (which represents a test or component document)
- **Options** object (which represents the Options dialog box)
- **Addins** collection (which represents a set of add-ins from the Add-in Manager dialog box)

You can also use the **Application** object to perform operations like loading add-ins, starting UFT, opening and saving tests or components, and closing UFT.

Each object returned by the **Application** object can return other objects, perform operations related to the object and retrieve and/or set properties associated with that object.

Every automation script begins with the creation of the UFT**Application** object. Creating this object does not start UFT. It simply provides an object from which you can access all other objects, methods and properties of the UFT automation object model.

> **Note:** You can also optionally specify a remote UFT computer on which to create the object (the computer on which to run the script). For details, see **Running Automation Programs on a Remote Computer** in the **Introduction** section of the *UFT Automation Object Model Reference* in the *UFT Help.*

UFT Automation Object Model Reference

Relevant for: GUI tests and components

The *HP UFT Automation Object Model Reference* is a Help file that provides detailed descriptions, syntax information, and examples for the objects, methods, and properties in the UFT automation object model.

You can open the *HP UFT Automation Object Model Reference* from:

- UFT program folder (**Start > All Programs > HP Software > HP Unified Functional Testing > Documentation > Unified Functional Testing Automation Reference**)

  > **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

- UFT Advanced References Help **(Help > HP UFT GUI Testing Advanced References Help > HP UFT Automation Object Model for GUI Testing)**

Generated Automation Scripts

Relevant for: GUI tests and components

The Properties pane of the Test Settings dialog box, the General node of the pane of the **GUITesting** tab in the Options dialog box, and the Object Identification dialog box each contain a **Generate Script** button. Clicking this button generates an automation script file (**.vbs**) containing the current settings from the corresponding dialog box.

You can run the generated script as is to open UFT with the exact dialog box configuration of the UFT application that generated the script, or you can copy and paste selected lines from the generated files into your own automation script.

For example, the generated script for the Options dialog box may look something like this:

```
Dim App 'As Application
Set App = CreateObject("QuickTest.Application")
App.Launch
App.Visible = True
App.Options.DisableVORecognition = False
App.Options.AutoGenerateWith = False
App.Options.WithGenerationLevel = 2
App.Options.TimeToActivateWinAfterPoint = 500
...
...
App.Options.WindowsApps.NonUniqueListItemRecordMode = "ByName"
App.Options.WindowsApps.RecordOwnerDrawnButtonAs = "PushButtons"
App.Folders.RemoveAll
```

For more details on the **Generate Script** button and the options available in the Options, Object Identification, and Test Settings dialog boxes, see "Configuring Object Identification" on page 1310, "UFT Global Options" on page 522, and "Settings for GUI Tests, GUI Business Components, and Application Areas" on page 580.

Tasks

How to Create a UFT Automation Script

Relevant for: GUI tests and components

This task describes when and how to create automation scripts, and includes the following steps:

- "Prerequisites" below
- "Create the Application object" below
- "Reference the type library - optional" on the next page
- "Write your automation script" on page 1163
- "Run your automation script" on page 1164

1. ### Prerequisites

   - **Decide whether to use UFT Automation Scripts**

     Creating a useful UFT automation script requires planning, design time, and testing. You must always weigh the initial investment with the time and human-resource savings you gain from automating potentially long or tedious tasks.

     Any UFT operation that you must perform many times in a row or must perform on a regular basis is a good candidate for a UFT automation script.

   - **Choose a language and development environment for designing and running Automation scripts**

     You can write your UFT automation scripts in any language and development environment that supports automation. For example, you can use: VBScript, JavaScript, Visual Basic, Visual C++, or Visual Studio .NET.

     For each language, there are a number of development environments available for designing and running your automation scripts.

2. ### Create the Application object

   The procedure for creating the **Application** object differs slightly from language to language. Below are some examples for creating the UFT**Application** object and starting UFT in visible mode, using different programming languages. For conceptual details on the **Application** object, see "Application Object" on page 1158.

Visual Basic

The example below can be used only after setting a reference to the type library. If you are not working in a development environment that allows referencing type libraries, create the **Application** object as described for VBScript below.

```
Dim qtApp As QuickTest.Application ' Declare the object
Set qtApp = New QuickTest.Application ' Create the object
qtApp.Launch ' Start QuickTest
qtApp.Visible = True ' Make it visible
```

VBScript

```
Dim qtApp
Set qtApp = CreateObject("QuickTest.Application")
qtApp.Launch 'Start QuickTest
qtApp.Visible = True ' Make it visible
```

JavaScript

```
// Create the application object
var qtApp = new ActiveXObject("QuickTest.Application");
qtApp.Launch(); // Start QuickTest
qtApp.Visible = true; // Make it visible
```

Visual C++

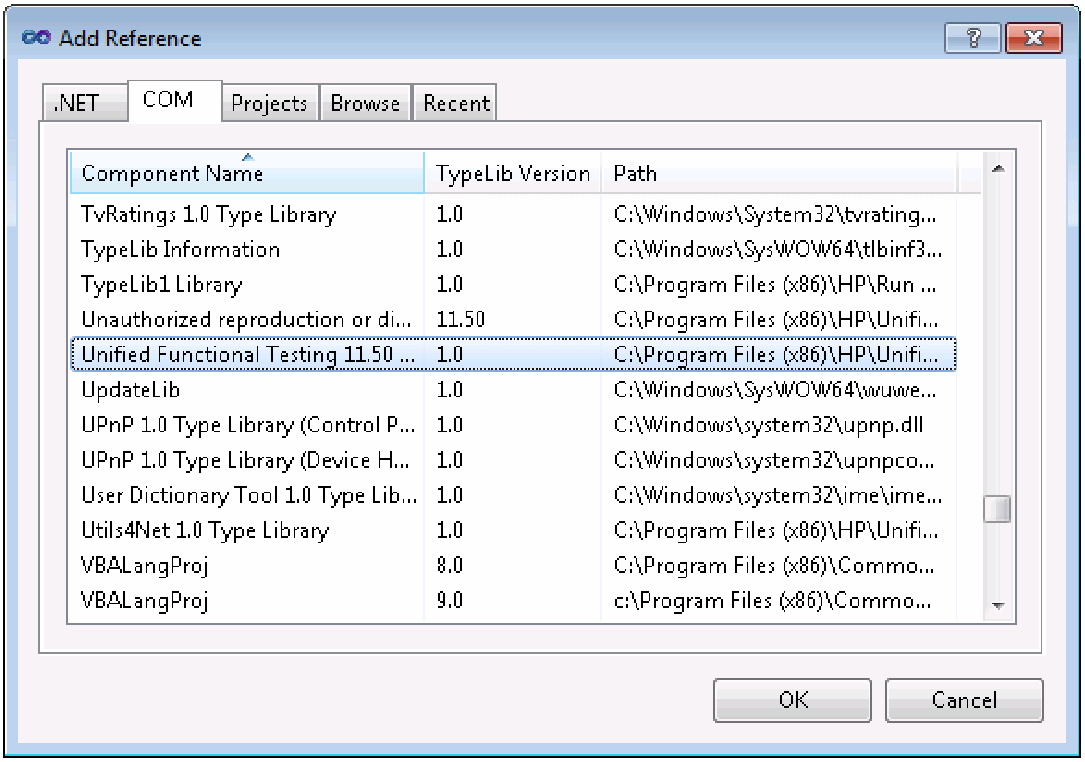
```
#import "QTObjectModel.dll" // Import the type library
// Declare the application pointer
QuickTest::_ApplicationPtr spApp;
// Create the application object
spApp.CreateInstance("QuickTest.Application");
spApp->Launch(); // Launch the application
spApp->Visible = VARIANT_TRUE; // Make it visible
```

3. Reference the type library - optional

Some development environments support referencing a type library. A **type library** is a binary file containing the description of the objects, interfaces, and other definitions of an object model.

If you choose a development environment that supports referencing a type library, you can take advantage of features like Microsoft IntelliSense, automatic statement completion, and status bar help tips while writing your script. The UFT automation object model supplies a type library file named **QTObjectModel.dll**. This file is stored in **<UFT installation folder>\bin**.

If you choose an environment that supports it, be sure to reference the UFT type library before you begin writing or running your automation script. For example, if you are working in Microsoft Visual Basic, select **Project > Add Reference** to open the **Add Reference** dialog box for your project. Then select **Unified Functional Testing<Version> Object Library** (where <**Version**> is the current installed version of the UFT automation type library).



Component Name	TypeLib Version	Path
TvRatings 1.0 Type Library	1.0	C:\Windows\System32\tvrating...
TypeLib Information	1.0	C:\Windows\SysWOW64\tlbinf3...
TypeLib1 Library	1.0	C:\Program Files (x86)\HP\Run ...
Unauthorized reproduction or di...	11.50	C:\Program Files (x86)\HP\Unifi...
Unified Functional Testing 11.50 ...	1.0	C:\Program Files (x86)\HP\Unifi...
UpdateLib	1.0	C:\Windows\SysWOW64\wuwe...
UPnP 1.0 Type Library (Control P...	1.0	C:\Windows\system32\upnp.dll
UPnP 1.0 Type Library (Device H...	1.0	C:\Windows\system32\upnpco...
User Dictionary Tool 1.0 Type Lib...	1.0	C:\Windows\system32\ime\ime...
Utils4Net 1.0 Type Library	1.0	C:\Program Files (x86)\HP\Unifi...
VBALangProj	8.0	C:\Program Files (x86)\Commo...
VBALangProj	9.0	c:\Program Files (x86)\Commo...

4. **Write your automation script**

The structure for your script depends on the goals of the script. You may perform a few operations before you start UFT such as retrieving the associated add-ins for a test or component, loading add-ins, and instructing UFT to open in visible mode.

After you perform these preparatory steps, if UFT is not already open on the computer, you can open UFT using the **Application.Launch** method. Most operations in your automation script are performed after the Launch method.

For details on the operations you can perform in an automation program, see *HP UFT Automation Object Model Reference* (**Help > HP UFT GUI Testing Advanced References Help > HP UFT Automation Object Model Reference**).

> **Tip:** You can generate automation scripts from UFT that contain the settings for the Test Settings dialog box, the GUI Testing tab in the Options dialog box, and the Object Identification dialog box as they are set on your computer. You can then run each generated script as is to instruct UFT to open on other computers with the exact dialog box

> configuration defined in the generated script, or you can copy and paste selected lines from the generated files into your own automation script. For details, see "Generated Automation Scripts " on page 1159.

When you finish performing the necessary operations, or you want to perform operations that require closing and restarting UFT (such as changing the set of loaded add-ins), use the **Application.Quit** method.

5. Run your automation script

There are several applications available for running automation scripts. You can also run automation scripts from the command line using Microsoft's Windows Script Host.

For example, you could use the following command line to run your automation script:

```
WScript.exe /E:VBSCRIPT myScript.vbs
```

For details on running automation scripts on a remote computer, see "How to Run Automation Scripts on a Remote Computer " below.

How to Run Automation Scripts on a Remote Computer

Relevant for: GUI tests and components

By default, when you create an Application object in your automation script, it is created on your local computer (using your local copy of UFT). You can also run automation scripts on a remote UFT computer. To do so, you must:

1. Set DCOM Configuration Properties on the Remote Computer

UFT automation enables UFT to act as a COM automation server. Therefore, to run a UFT automation script on a remote computer, you must ensure that the DCOM configuration properties for that computer give you the proper permissions to launch and configure the UFT COM server.

The procedure below describes the steps you need to perform on the remote computer to enable your automation script to run on that computer. Note that the DCOM Configuration Property the appearance and names of the dialog boxes and options mentioned here may vary depending on the computer's operating system.

To enable automation scripts to access a Unified Functional Testing computer remotely:

a. On the computer where you want to run the automation script, select **Start > Run**. The Run dialog box opens.

b. Enter **dcomcnfg** and click **OK**. The Distributed COM Configuration Properties dialog box or the Component Services window opens (depending on your operating system) and displays the list of COM applications available on the computer.

c. Select **QuickTest Professional Automation** from the DCOM Config list and open the Properties dialog box for the application. (Click the **Properties** button or right-click and select **Properties**, depending on your operating system.)

d. In the QuickTest Professional Automation Properties dialog box, click the **Security** tab.

e. In the launch permissions section, select the custom option and click **Edit**.

f. Use the **Add** and **Remove** options to select the network users or groups for which you want to allow or deny permission to launch UFT via an automation script. When you are finished, click **OK** to save your settings.

g. Repeat the previous two steps for the configuration permissions section to select the users or groups who can modify UFT configuration options via an automation script.

h. In the QuickTest Professional Automation Properties dialog box, click the **Identity** tab and select the **interactive user** option.

i. Click **OK** to save the QuickTest Professional Automation Properties settings.

j. Click **OK** to close the Distributed COM Configuration Properties dialog box or the Component Services window.

2. **Create an Application Object on the Remote Computer**

After you set the necessary DCOM Configuration settings for a remote computer, you can specify that computer in your automation script.

In VBScript, you do this by specifying the computer name as the optional location argument of the CreateObject function. The computer name should be the same as the computer name portion of a share name. For example, to run an automation script on a computer called MyServer, you could write:

```
Dim qtApp
Set qtApp = CreateObject("QuickTest.Application", "MyServer")
```

For details on the syntax for specifying the remote computer in another language you are using, see the documentation included with your development environment or the general documentation for the programming language.

Part 7: GUI Testing: Test Objects, Checkpoints and Output Values

Chapter 42: The Test Object Model

Relevant for: GUI tests and components

This chapter includes:

Concepts

Test Object Model - Overview

Relevant for: GUI tests and components

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

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

This section includes:

How UFT Learns Objects

Relevant for: GUI tests and components

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

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

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

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

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

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

UFT uses a very similar method when it learns objects.

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

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

Insight Image-Based Identification

To learn an object based on its properties, UFT must recognize the technology in which the object was developed, and retrieve its properties via the object's programmatic interface. When this is not possible, you can use UFT's Insight option, to learn the object based on its appearance.

When using Insight, UFT stores the object as an InsightObject test object, and captures an image of the object to use as its main identification property. If the application includes additional similar objects, and the image is not sufficient to create a unique description of the object, UFT adds an ordinal identifier, in the same way as it would when learning an object based on its properties.

How UFT Identifies Objects During a Run Session

Relevant for: GUI tests and components

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

Suppose as a continuation to the experiment (described in "How UFT Applies the Test Object Model Concept" on page 1171), Alex is now asked to identify the same "item" he initially identified but in a new, yet similar environment.

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

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

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

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

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

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

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

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

UFT uses a very similar process of elimination with its **Smart Identification** mechanism to identify an object, even when the learned description is no longer accurate. Even if the values of your identification properties change, UFT maintains the reusability of your test or component by identifying the object using Smart Identification. For details on Smart Identification, see "Configuring Object Identification" on page 1310.

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

For a flowchart illustrating the general object identification process, see "Object Identification Process Workflow" on page 1188.

Using Insight to Identify an Object During a Run Session

When using Insight, UFT searches for an object that matches the image stored with the test object. It also uses the ordinal identifier, if one was stored with the object, or any visual relation identifiers that you defined. UFT searches for the matching object within the Insight object's parent test object. You can help focus the search on a smaller area by creating smaller parent objects and building a hierarchy of Insight test objects in your object repository.

Image comparison can be time consuming, especially if there are many similar objects within the same parent. Therefore, steps with Insight objects might take longer to run than steps with UFT test objects that use object properties for identification.

UFT's image matching algorithm allows for some variation, enabling UFT to recognize the object even if it changed slightly. However, the algorithm is not based on object properties, and therefore does not use the smart identification mechanism.

How UFT Applies the Test Object Model Concept

Relevant for: GUI tests and components

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

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

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

When UFT learns an object in your application, it adds the corresponding test object to an **object repository**, which is a storehouse for objects. You can add test objects to an object repository in several ways. For example, you can use the Navigate and Learn option, add test objects manually, or perform an operation on your application while recording. For details on object repositories, see "Managing Test Objects in Object Repositories" on page 1198, "Shared Object Repositories" on page 1285, and "Test Creation - Keyword-Driven Methodology" on page 820.

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

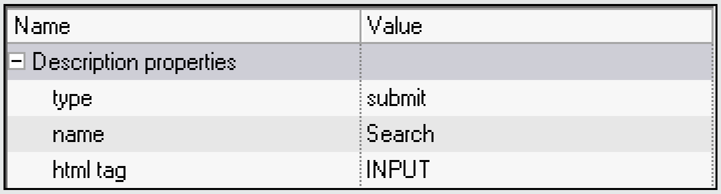
- Identifies the UFT test object class that represents the learned object and creates the appropriate test object.
- Reads the current value of the object's properties in your application and stores the list of **identification properties** and values with the test object.
- Chooses a unique name for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

Example

Suppose you add a **Search** button with the following HTML source code:

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

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



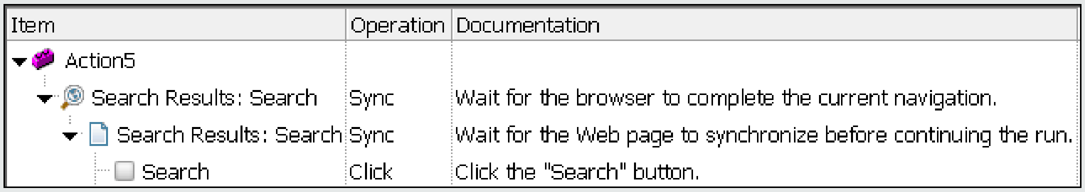
Name	Value
⊟ Description properties	
type	submit
name	Search
html tag	INPUT

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

In an action: UFT displays your step in the Editor like this:

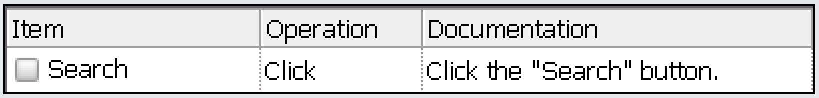
```
Browser("Search Results: Search").Page("Search Results:
  Search").WebButton("Search").Click
```

and in the Keyword View like this:



Item	Operation	Documentation
Action5		
Search Results: Search	Sync	Wait for the browser to complete the current navigation.
Search Results: Search	Sync	Wait for the Web page to synchronize before continuing the run.
□ Search	Click	Click the "Search" button.

In a component: The hierarchy is not displayed, and the step is displayed like this:



Item	Operation	Documentation
□ Search	Click	Click the "Search" button.

When you run a test or 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). The list of test objects and their properties and values are stored in the object repository. In the above example, UFT would search in the object repository during the run session for the WebButton object with the name Search to look up its description. Based on the description it finds, UFT would then look for a WebButton object in the application with the HTML tag INPUT, of type submit, with the value Search. When it finds the object, it performs the **Click** method on it.

This section also includes:

Test Object Descriptions

Relevant for: GUI tests and components

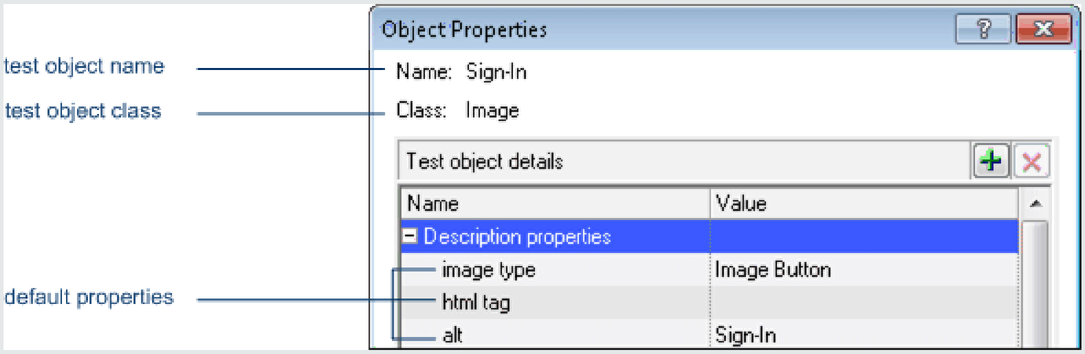
For each test object class, UFT learns a set of identification properties when it learns an object, and selects a sub-set of these properties to serve as a unique object description. UFT then uses this description to identify the object when it runs a test or component.

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

You can configure the mandatory, assistive, and ordinal identifier properties that UFT uses to learn the descriptions of the objects in your application, and you can enable and configure the Smart Identification mechanism. For details, see "Configuring Object Identification" on page 1310.

Example

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



Name: Sign-In

Class: Image

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

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

Insight Test Object Descriptions

When using Insight, UFT learns objects based on their appearance instead of retrieving properties from the objects. For the description property, UFT stores an image of the object, which can be used later to identify the object. If parts of the object do not always look the same, you can instruct UFT to ignore those areas when it uses the image to identify the object.

If necessary, UFT can also use an ordinal identifier to create a unique description for the object. Other aspects of object configuration, such as mandatory and assistive properties, and smart identification, are not relevant for Insight test objects.

After UFT creates a description for an Insight test object, you can add visual relation identifiers to improve identification of the object. If necessary, you can also add the **similarity** identification property to the test object description. This property is a percentage that specifies how similar a control in the application has to be to the test object image for it to be considered a match.

UFT Test Object Hierarchy

Relevant for: GUI tests and components

The UFT test object hierarchy comprises one or more levels of test objects. The top level object may represent a window, dialog box, or browser type object, depending on the environment. The actual object on which you perform an operation may be learned as a top level object, a second level object, for example, Window.WinToolbar, or a third level object, for example, Browser.Page.WebButton.

In some cases, even though the object in your application may be embedded in several levels of objects, the hierarchy does not include these objects. For example, if a WebButton object in your application is actually contained in several nested WebTable objects, which are all contained within a Browser and Page, the learned object hierarchy is only Browser.Page.WebButton.

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

For details on the UFT object hierarchy for specific environments, see the relevant section in the *HP Unified Functional Testing Add-ins Guide*.

Properties and Operations for Test Objects and Run-Time Objects

Relevant for: GUI tests and components

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

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

As you add steps to your test or component, you specify which operation to perform on each test object. If you record steps, UFT records the relevant operation as it is performed on an object. During a run session, UFT performs the specified test object operation on the run-time object.

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

For test objects

- You can retrieve or modify identification property values manually while designing your test or component, or you can use **SetTOProperty** statements during a run session (via an operation defined in a function library).

  For details, see "Managing Test Objects in Object Repositories" on page 1198, "Retrieving and Setting Identification Property Values" on page 1007, and the *HP UFT Object Model Reference for GUI Testing*.

- You can use regular expressions in function libraries to identify property values based on conditions or patterns you define.

For details, see "Regular Expressions Overview" on page 318.

- You can parameterize identification property values with data table parameters so that a different value is used during each iteration of the test.

  For details, see "Parameterizing Object Values" on page 1526.

- You can view or modify the identification property values that are stored with your test or component in the Object Properties or Object Repository window.

  For details, see "Specifying or Modifying Property Values" on page 1205.

- You can view the current identification property values of any visible object using the Properties tab of the Object Spy.

  For details, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 1184.

- You can view the syntax of the test object operations of any visible object using the Operations tab of the Object Spy. For details, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 1184.

For run-time objects

- You can view the syntax of the native operations of any visible object using the Operations tab of the Object Spy. For details, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 1184.

- You can retrieve native property values from the run-time object during the run session by adding **GetROProperty** statements. For details, see "Retrieving and Setting Identification Property Values" on page 1007.

- If the available test object operations and identification properties do not provide the functionality you need, you can access the internal operations and native properties of the run-time object using the **Object** property. You can also use the **attribute** object property to identify Web objects in your application according to user-defined properties. For details, see "Native Properties and Operations" on page 1008 and the *HP UFT Object Model Reference for GUI Testing*.

- When using Insight test objects, which UFT recognizes in the application based on the object's appearance, UFT does not use the object's programming interface and therefore native operations and properties are not relevant.

Identifying Objects Using Insight

Relevant for: GUI tests and components

UFT can use Insight, which is an image-based identification ability, to recognize objects in your application based on what they look like, instead of using properties that are part of their design. This can be useful if you are working with an application whose technology is not supported by UFT, or with an application running on a remote computer.

When using Insight, UFT stores an image of the object with the Insight test object (along with ordinal identifiers, if necessary), and uses this image as the main description property to identify the object in the application. For more details, see "Insight Test Object Descriptions" on page 1173.

You can create InsightObject test objects during recording sessions, or when adding test objects to an object repository manually. When adding objects to an object repository, you can add an object directly from the application, or even from a picture of the object displayed on your screen.

> **Note: Dual monitor support.** Insight is supported only on the primary monitor. Therefore, if you are working with dual monitors, make sure that your application is visible on the primary monitor when you use UFT for Insight test objects.

For details, see:

- "How to Record a Test or Component Using Insight Recording" on page 860
- "How to Add an Insight Object to the Object Repository" on page 1214

In the object repository, you can edit the object's properties, such as its name and image, and the default location to click in the control when performing test object methods. You can also define visual relation identifiers to improve UFT's ability to accurately identify the object.

Insight Test Object Hierarchy

The Insight object is always added to the object repository as a child of the test object that represents its containing application, such as a Window or Browser object. (The new object's parent object is also added if it does not already exist in the object repository.)

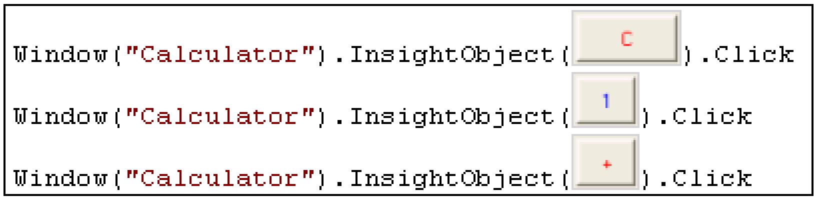
You can manually move the test object in the Object Repository window or the Object Repository Manager. If you place it under another test object, then UFT searches for the object in the application only within its parent test object. If you move the Insight test object to be a top-level object, then UFT searches for the object anywhere on the screen.

Steps with Insight Test Objects

You create steps using Insight test objects in much the same way as you would with other types of test objects.

For details on the methods and properties supported by Insight test objects, see the **Insight** section of the *HP UFT Object Model Reference for GUI Testing*.

In the Editor, the test object image is displayed in the step instead of the test object name. When you hold the cursor over the image, an enlarged view of the image is displayed. You can double-click the image to open the object repository with the InsightObject selected.



```
Window("Calculator").InsightObject( C ).Click
Window("Calculator").InsightObject( 1 ).Click
Window("Calculator").InsightObject( + ).Click
```

If you want the Editor to show the test object names instead of the images, clear the **Show test object image in steps** option in the Insight pane of the Options dialog box (**Tools > Options >**

GUI Testing tab **> Insight** node). For details, see "Insight Pane (Options Dialog Box > GUI Testing Tab)" on page 561.

When running steps on Insight objects, UFT can show a visual representation of the steps being performed on the application. For example, a small circle is displayed for mouse clicks, and a line is drawn to show the path of a drag operation. You can enable this option in the Insight pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Insight** node).

During the run session, UFT searches for an object that matches the image stored with the test object. Image comparison can be time consuming, especially if there are many similar objects within the same parent. Therefore, steps with Insight objects might take longer to run than steps with UFT test objects that use object properties for identification.

Object Repository Types - Overview

Relevant for: GUI actions and components

Objects can be stored in two types of object repositories—a shared object repository and a local object repository.

A **shared object repository** stores objects in a file that can be accessed by multiple tests or components (via their application areas) (in read-only mode). You can use the same shared object repository for multiple actions or components. You can also use multiple object repositories for each action or component.

A **local object repository** stores objects in a file that is associated with one specific action or component, so that only that action or component can access the stored objects.

Tests or components always use the object repositories that are specified in the Associated Repositories tab of the Action Properties dialog box or in the Associate Repositories dialog box of the application area with which the component is associated. Shared object repositories are read-only when accessed from tests or components; you edit them using the Object Repository Manager.

You perform many object repository-related tasks either in the Object Repository window or in the Object Repository Manager. To view a list of the tasks that you can perform in each, see "Comparison of Object Repository Window and Object Repository Manager " on page 1186.

Planning where to store objects

When you plan and create tests or components, you must consider how you want to store their test objects. You can:

- Store the objects for each action or component in its corresponding local object repository.
- Store the objects in one or more shared object repositories. By storing objects in shared object repositories and associating these repositories with your actions or component's application areas, you enable multiple actions and components to use the objects. Use a combination of objects from your local and shared object repositories, according to your needs.

- Transfer local objects to a shared object repository, if required. This reduces maintenance and enhances the reusability of your tests and components because it enables you to maintain the objects in a single, shared location instead of multiple locations. For details, see "Deciding Whether to Use Local or Shared Object Repositories " on the next page.

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

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

- If an object with the same name is located in both the local object repository and in a shared object repository associated with the same action or component, the local object definition is used.
- If more than one shared object repository is associated with the same action or component, the object definition is used from the first occurrence of the object, according to the order in which the shared object repositories are associated with the action or component.

  For details on associating shared object repositories, see:

  - **For tests:**"Associate Repositories Dialog Box" on page 1266
  - **For components:** "Object Repositories Pane (Application Area)" on page 2105

Note for users of previous QuickTest versions

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

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

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

Deciding Whether to Use Local or Shared Object Repositories

Relevant for: GUI actions and components

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

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

Use this object repository type...	In these situations...
local object repository	• You are creating single-action tests. • You are creating simple tests or components, especially under the following conditions: ▪ You have only one, or very few, tests or components that correspond to a given application, interface, or set of objects. ▪ You do not expect to frequently modify object properties. ▪ You are new to using UFT. You can record and run tests or components without creating, choosing, or modifying shared object repositories because all objects are automatically saved in a local object repository that can be accessed by its corresponding action or component. **See also:** "Local Object Repository - Overview" on the next page

Use this object repository type...	In these situations...
shared object repository	- You are creating tests or components using keyword-driven methodologies (not by recording). - You have several tests or components that test elements of the same application, interface, or set of objects. - You often work with multi-action tests and regularly use the **Insert Copy of Action** and **Insert Call to Action** options. - You expect the object properties in your application to change from time to time and/or you regularly need to update or modify object properties. - If you are familiar with testing, it is probably most efficient to save objects in a shared object repository. In this way, you can use the same shared object repository for multiple actions or components—if they use the same objects. Object information that applies to many actions or components is kept in one central location. When the objects in your application change, you can update them in one location for all the actions and components that use this shared object repository. **See also:** "Shared Object Repository - Overview" on the next page

This section also includes:

Local Object Repository - Overview

Relevant for: GUI actions and components

When you use a local object repository, UFT uses a separate object repository for each action or component. (You can also use one or more shared object repositories if needed. For details, see "Shared Object Repositories Overview" on page 1286.) The local object repository is fully editable from within its action or component.

When working with a local object repository:

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

  UFT adds all new objects to the local object repository even if one or more shared object repositories are already associated with the action or component. (This assumes that a test object with the same description does not already exist in one of the associated shared object repositories).
- If a child object is added to a local object repository, and its parents are in a shared object repository, its parents are automatically added to the local object repository.
- Every time you create a new action or component, UFT creates a new, corresponding local object repository and adds test objects to the repository as it learn them.
- If UFT learns the same object in your application in two different actions or components, the test object is stored as a separate test object in each of the local object repositories.
- When you save your test or component, its local object repositories (or repository, respectively) are automatically saved with the test or component. The local object repository is not accessible as a separate file (unlike the shared object repository).

Shared Object Repository - Overview

Relevant for: GUI actions and components

When you use shared object repositories, UFT uses the shared object repositories you specify for the selected action or component's application area. You can use one or more shared object repositories. (You can also save some objects in a local object repository for each action or component if you need to access them only from the specific action or component. For details, see "Local Object Repository - Overview" on the previous page.)

After you begin creating your test or component, you can specify additional shared object repositories. You can also create new ones and associate them with your action or component. Before running the test or component, you must ensure that the object repositories being used contain all of the test objects used in the test or component steps. Otherwise, the test or component may fail. For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199.

You modify a shared object repository using the Object Repository Manager. For details, see "Shared Object Repositories" on page 1285.

When working with a shared object repository:

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

You can export objects from the local object repository to a shared object repository. You can also export the local object repository and replace it with a shared object repository. This enables you to make the local objects accessible to other actions or components. For details, see "Exporting Local Objects to a Shared Object Repository" on page 1260.

You can also merge objects from the local object repository directly to a shared object repository that is associated with the same action or component. This can help reduce maintenance since you can maintain the objects in a single shared location, instead of multiple locations. For details, see "When to Merge Shared Object Repositories" on page 1352.

Tasks

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

Relevant for: GUI tests and components

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

This task includes the following steps:

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

1. Prerequisites

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

2. Open the Object Spy dialog box

For details, see the **To Access** section of the "Object Spy Dialog Box " (described on page 1189).

3. Select the details you want to view for the object

In the "Object Spy Dialog Box " (described on page 1189), select the **Properties** or **Operations** tab, and then select the radio button to view the **Native** or **Identification** properties. For details about using the run-time object's operations or retrieving the values of its properties, see "Retrieving and Setting Identification Property Values" on page 1007 and "Native Properties and Operations" on page 1008.

4. Resize the Object Spy dialog box - Optional

This is useful if the objects on which you want to spy have a deep hierarchy or long property names and values, as it enables you to view all of the information without scrolling. For details, see "Object Spy Dialog Box " on page 1189.

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

In the Object Spy, click the pointing hand button and then mouse over or click an object in your application. In most environments, as you mouse over objects in your application, the Object Spy highlights the object it identifies and displays the relevant information in the Object Spy, according to the options you selected in the Object Spy dialog box. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.

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

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

6. Add selected objects to the object repository - Optional

After clicking an object you can add it (or any other object in the **Object hierarchy** tree) to the object repository using the **Add to Repository** button . For details, see "Object Spy Dialog Box " on page 1189.

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

Reference

Comparison of Object Repository Window and Object Repository Manager

Relevant for: GUI tests and components

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

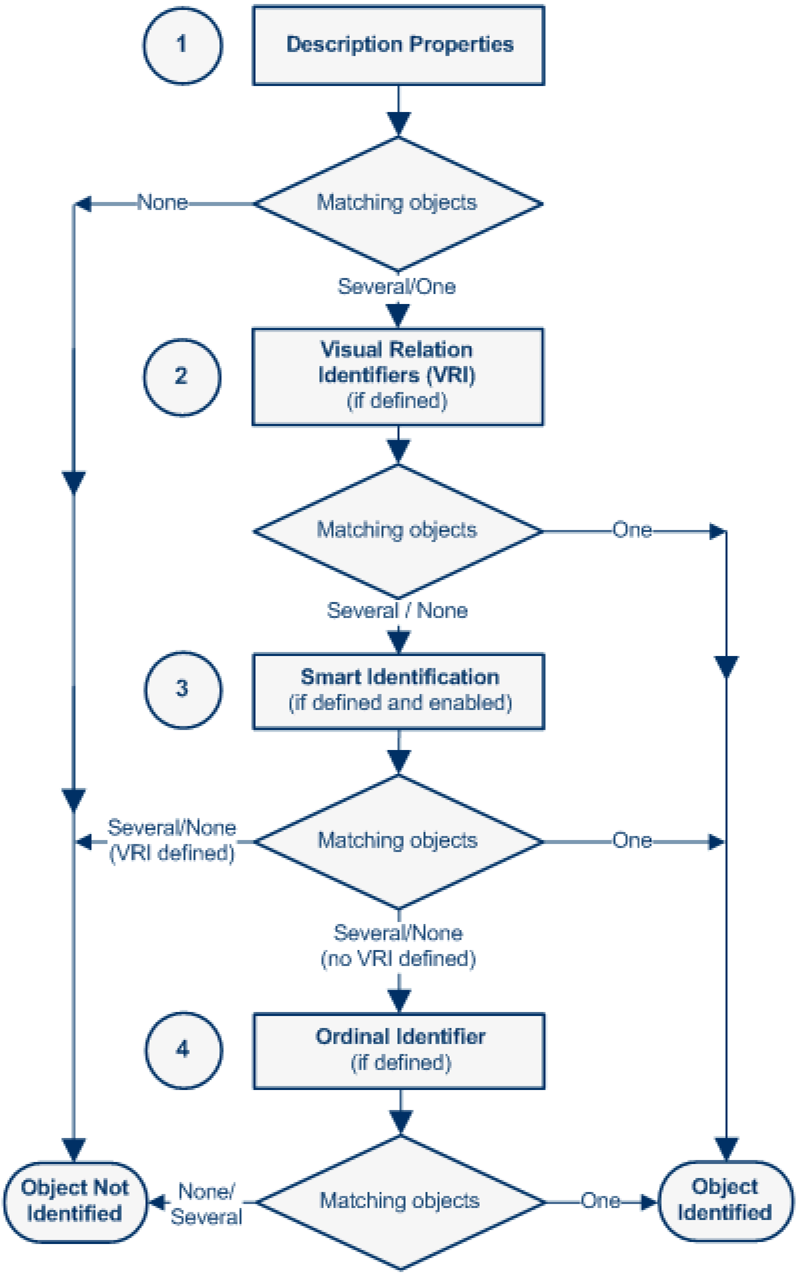
Functionality	Object Repository window	Object Repository Manager
"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199	✓	✓
"Copying, Pasting, Moving, or Deleting Objects in the Object Repository "on page 1217	✓	✓
"Highlighting an Object in Your Application" on page 1203	✓	✓
"Locating a Test Object in the Object Repository" on page 1204	✓	✓
"Specifying or Modifying Property Values" on page 1205	✓	✓
"Updating Identification Properties from an Object in Your Application" on page 1206	✓	✓
"Restoring Default Mandatory Properties for a Test Object" on page 1206	✓	✓
"Renaming Test Objects " on page 1206	✓	✓
"Adding Properties to a Test Object Description" on page 1208	✓	✓
"Defining New Identification Properties" on page 1208	✓	✓
"Removing Properties from a Test Object Description" on page 1224	✓	✓
"Exporting Local Objects to a Shared Object Repository" on page 1260	✓	✗
"Copying Objects to the Local Object Repository" on page 1261	✓	✗
"Managing Shared Object Repositories" on page 1291 (includes creating, opening, saving, closing, and enabling editing functionality)	✗	✓

Functionality	Object Repository window	Object Repository Manager
"Adding a Test Object to an Object Repository" on page 1211	✓	✓
"Adding Test Objects Using the Navigate and Learn Option" on page 1295	✗	✓
"Working with Repository Parameters" on page 1287	✗	✓
"Merging Two Shared Object Repositories" on page 1356	✗	✓
"Importing and Exporting Shared Object Repositories Using XML" on page 1287	✗	✓

Object Identification Process Workflow

Relevant for: GUI tests and components

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



Note for Web-based objects:

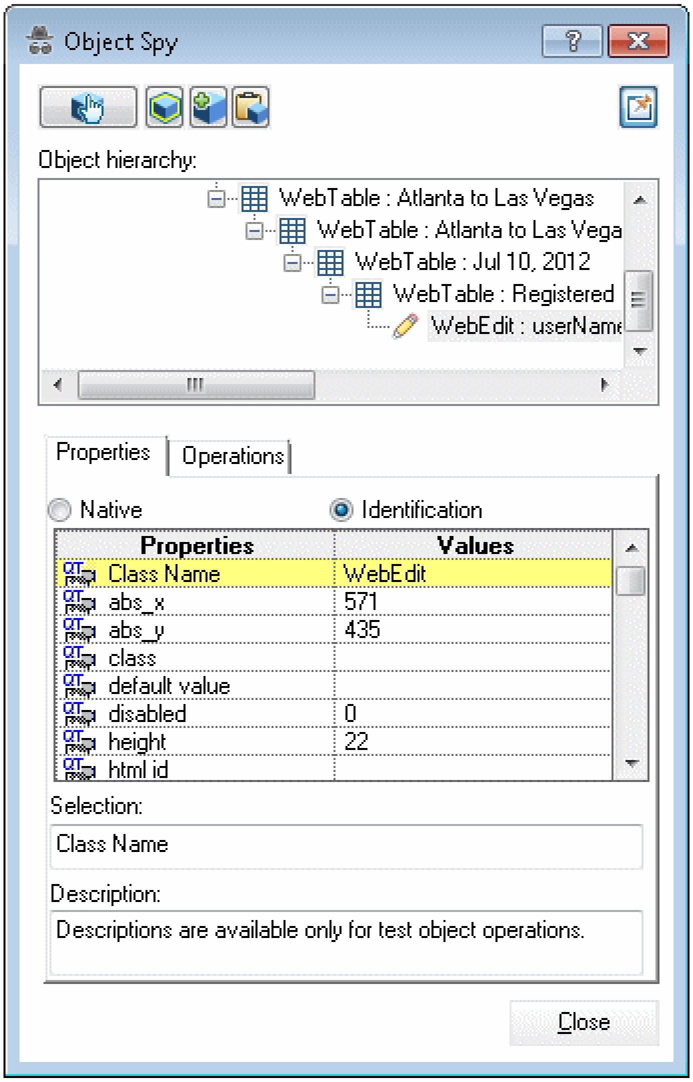
- If you defined Web object identifiers (such as XPath/CSS properties) for these test objects, they are used before the description properties. If one or more objects are found, UFT continues to identify the object using the description properties. For details, see the section on Web Object Identifiers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

Object Spy Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and operations of the test object that UFT uses to represent that object.

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



Object hierarchy:

```
WebTable : Atlanta to Las Vegas
  WebTable : Atlanta to Las Vega
    WebTable : Jul 10, 2012
      WebTable : Registered
        WebEdit : userName
```

Properties | Operations

○ Native ● Identification

Properties	Values
Class Name	WebEdit
abs_x	571
abs_y	435
class	
default value	
disabled	0
height	22
html id	

Selection:

Class Name

Description:

Descriptions are available only for test object operations.

To access	1. Do one of the following: ▪ Ensure that a GUI test or action is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or action node. 2. Use one of the following: ▪ Select **Tools > Object Spy**. ▪ Click the **Object Spy** toolbar button . ▪ In the Record toolbar, click the **Object Spy** button during a recording session. Available from: • "UFT Main Window Overview" (described on page 78) • "Object Repository Window" (described on page 1274) • "Object Repository Manager Main Window" (described on page 1297) • "Record Toolbar" (described on page 866)
Relevant tasks	"How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 1184
See also	"Properties and Operations for Test Objects and Run-Time Objects" on page 1175

User interface elements are described below:

Option	Description
☝	**Pointing Hand.** Turns the mouse pointer into a pointing hand. Use the pointing hand to highlight or click the object whose properties and/or operations you want to view. • As you move the pointing hand over the objects in the application, the objects are highlighted in the application (in some environments), and their details are displayed in the Object Spy dialog box. • To capture information about a particular object and its parent objects in the Object Spy, click on the object in the application. **See also:** "Tips for Using the Pointing Hand" on page 1196.

Option	Description
📌	**Keep on Top.** Keeps the Object Spy dialog box in view while spying on an object in your application. **Note:** When the **Keep on Top** button is not pressed, the Object Spy dialog box may be hidden on your screen behind your application. To view the Object Spy dialog box, press the left **CTRL** key and arrange the windows as needed.
⬡	**Highlight in Application.** Highlights the object in the application that corresponds to the test object currently selected in the Object Spy.
⊕	**Add Object to Repository.** Adds the test Object selected in the Spy hierarchy tree to the object repository. • If the object is successfully added to the object repository, the repository icon 🗃 is added to the bottom of the test object icon. • When the Object Spy dialog box is opened from UFT or the Object Repository Window, the object is added to the local object repository of the active action or component. • When the Object Spy dialog box is opened from the Object Repository Manager, the object is added to the active object repository. This option is disabled when there is no open repository.

Option	Description

📋	**Copy Identification Properties to Clipboard**. Copies all of the properties and values for the object currently selected in the **Object hierarchy** tree. You can paste the copied data from the Clipboard into any document.
	Enabled only when the **Identification** radio button is selected in the **Properties** tab.
	The copied properties and values are formatted in standard programmatic description syntax with line breaks between each property-value pair. For example:
	`"Class Name:=Image",`
	`"abs_x:=585",`
	`"abs_y:=573",`
	`"alt:=Specials",`
	`....`
	This option is useful if you want to:
	- Compare the properties and values of two objects in your application.
	- Copy the relevant strings when creating programmatic descriptions.
	For more details on programmatic descriptions, see "Programmatic Descriptions" on page 995.

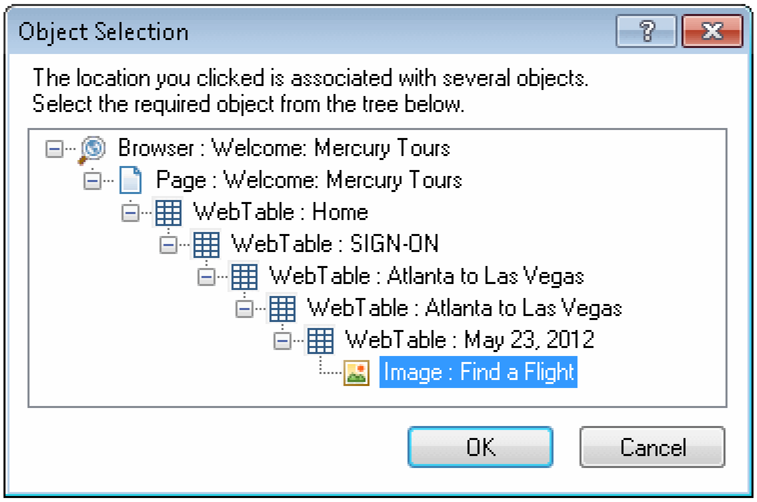
Object hierarchy	Displays the hierarchy of test objects that are related to the object you selected in your application. - If an object in the hierarchy already exists in a repository associated with the active action or component, a repository icon 🗄 is displayed in the lower-right corner of the object's icon. - To view properties, values, or operations for another test object in the **Object hierarchy** tree, select that test object in the tree. - While an object is highlighted in the application, test object classes are displayed in the **Object hierarchy** tree, but test object names are not. Test object names (such as Atlanta to Las Vegas and Specials in the image shown above) are displayed only after clicking the object to capture the information in the Object Spy.

Option	Description
Properties tab	Select the **Native** or **Identification** radio button to display the native properties and their values, or the test object identification properties and values of the run-time object associated with the selected test object in the **Object hierarchy** tree. • **Properties.** Displays the identification or native property names for the test object that is currently selected in the **Object hierarchy** tree. • **Values.** Displays the identification or native property values of the relevant object in the application.
Operations tab	Select the **Native** or **Test Object** radio button to display the native operations or test object operations, and their corresponding syntax, for the test object that is currently selected in the Object Spy test **Object hierarchy** tree, or the run-time object associated with it.
Selection	The content in this box differs depending on which tab is displayed above it. **Properties tab:** Displays the property name or value that was most recently clicked. **Operations tab:** Displays the syntax of the most recently clicked operation. **Tip:** To copy the text that is displayed in this box to the Clipboard, highlight the text and press Ctrl+C or right-click the highlighted text and select **Copy**.
Description	Provides a description of the most recently clicked operation, when available.

Object Selection Dialog Box

Relevant for: GUI tests and components

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



The location you clicked is associated with several objects.
Select the required object from the tree below.

- Browser : Welcome: Mercury Tours
  - Page : Welcome: Mercury Tours
    - WebTable : Home
      - WebTable : SIGN-ON
        - WebTable : Atlanta to Las Vegas
          - WebTable : Atlanta to Las Vegas
            - WebTable : May 23, 2012
              - Image : Find a Flight

To access	1. Do one of the following: ▪ Ensure that a GUI test or action is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test or action node. 2. Use one of the following: ▪ After performing an operation that changes the pointer to a pointing hand, if you point to a location that is associated with more than one object, the Object Selection dialog box opens. ▪ **For tests:** When you right-click a location in the Active Screen to perform an operation that requires selecting an object, if the location is associated with more than one object, the Object Selection dialog box opens.
See also	"Tips for Using the Pointing Hand" on the next page

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

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

Tips for Using the Pointing Hand

Relevant for: GUI tests and components

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

This section provides tips for working with the pointing hand.

- You can hold the left **CTRL** key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in UFT or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.

- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the GUI Testing tab in the Options dialog box (**Tools > Options > GUI Testing** tab > **General** node). For details, see "General Pane (Options Dialog Box > GUI Testing Tab)" on page 536.

- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left **CTRL** key and arrange the windows as needed.

- If the window containing the object you want to select is minimized, you can display it by holding the left **CTRL** key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

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

Examples of pointing hand usage

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

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

Troubleshooting and Limitations - Object Spy

Relevant for: GUI tests and components

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

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

- The Object Spy does not support Insight test objects.

Chapter 43: Managing Test Objects in Object Repositories

Relevant for: GUI tests an components

This chapter includes:

Concepts

Adding and Deleting Test Objects in a Local or Shared Object Repository

Available from:

- Object Repository window for the local object repository
- Object Repository Manager for shared object repositories

Relevant for: GUI tests and components

When you create an object repository for your keyword-driven testing infrastructure, you can add test objects to it in various ways, for example:

- You can use the **Navigate and Learn** option to add objects to a shared object repository according to your defined filter.
- You can record an action or component to add each object on which you perform an operation to the local object repository. (This occurs for objects that do not already exist in an associated shared object repository.)
- You can add test objects to the local object repository while editing your test or component. For example, for tests and scripted components, you can add tests objects from the Active Screen.

When you add test objects to an object repository, you can choose to add only a selected test object, to add all test objects of a certain type (such as all button objects), or to add all test objects of a specific class (such as all **WebButton** objects).

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

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

If needed, you can merge test objects from the local object repository into a shared object repository. For details, see "Object Repository Merge Tool" on page 1351.

You can also add test objects to a shared object repository while navigating through your application. For details, see "Navigate and Learn Toolbar" on page 1302.

This section also includes:

- "Adding Insight Test Objects" below
- "Adding Test Objects to the Local Object Repository from the Active Screen (GUI actions and scripted components only)" below
- "Defining New Test Objects" on the next page
- "Deleting Objects in the Object Repository" on the next page

Adding Insight Test Objects

Use Insight test objects in UFT if you want UFT to recognize objects based on what they look like, instead of based on their native properties. For details, see "Identifying Objects Using Insight" on page 1176.

When you add an Insight test object, UFT stores an image of the object together with the test object, and later uses this image to identify the object in the application.

When you add an Insight test object, the following learning modes are available:

- **Automatic.** When using this mode, you click on the control in the application, and UFT detects the borders of the control, and takes a snapshot of it. This mode is the faster mode and should be satisfactory in most cases.
- **Manual.** When using this mode, you take a snapshot of the control in the application, manually specifying the borders of the control. You can use this mode in cases where the automatic mode does not correctly detect the borders of the control. For example, when it selects an area of the application which is much larger than the control.

In both cases, UFT saves an image of the control (used to identify the control), as well as a snapshot of the screen containing the control. After UFT takes the snapshot, you can adjust the image used to identify the test object.

> **Note:** Insight test objects require more disk space than other test objects, because of the test object images and the snapshots stored with the test objects. To control the amount of space used, you can limit the size of the snapshot in the "Insight Pane (Options Dialog Box > GUI Testing Tab)".
>
> In addition, after you add all of the relevant test objects and finish modifying them to your satisfaction in the object repository, you can delete all of the snapshots to reduce the amount of disk space used.

For task details, see "How to Add an Insight Object to the Object Repository" on page 1214.

Adding Test Objects to the Local Object Repository from the Active Screen (GUI actions and scripted components only)

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

To add test objects to the object repository using the Active Screen, the Active Screen must contain information for the object you want to add. You can control how much information is captured in the Active Screen in the Active Screen node of the Options dialog box (**Tools > Options > GUI Testing** tab **> Active Screen** node). For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

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

Defining New Test Objects

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

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

When you manually define a new object in the object repository, the object is added to the local object repository and can be used only by the current action or component. If you want to add the object to the shared object repository so that it can be used in multiple actions or components, you must add it using the Object Repository Manager. For details, see "Shared Object Repositories" on page 1285.

After you define a new test object, you can modify the test object description at any time. For example, you may need to update the test object description if the properties of the object in your application do not match the test object description that you defined, or if an object was updated in your application. For details, see "Updating Identification Properties from an Object in Your Application" on page 1206.

Deleting Objects in the Object Repository

When you remove a step from your action or component, its corresponding test object remains in the object repository. You can delete the test object if it is not in use elsewhere.

For example, if you are working with a local object repository, make sure that the test object is not used in any other steps within that action or component. If you are working with a shared object repository, confirm that the test object is not used in any other action or component using that shared object repository. Also, confirm that the test object you are deleting is not used in a visual relation identifier for another test object.

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

> **Note:** If your action or component contains references to an object that you deleted from the

object repository, your run session will fail.

Guidelines for Copying, Pasting, and Moving Objects

Relevant for: GUI tests and components

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

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

Locating Objects

Available from:

- Object Repository window for the local object repository
- Object Repository Manager for shared object repositories

Relevant for: GUI tests and components

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

For task details, see "How to Locate an Object in an Object Repository" on page 1219.

Note: When locating test objects, UFT also takes into account the visual relation identifier property, if defined for that test object. For details, see "Visual Relation Identifiers" on page 1209.

This section also includes:

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

Finding Objects in an Object Repository

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

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

Highlighting an Object in Your Application

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

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

Locating a Test Object in the Object Repository

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

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

Maintaining Identification Properties

Available from:

- Object Repository window for the local object repository
- Object Repository Manager for shared object repositories

Relevant for: GUI tests and components

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

If you are using an Insight object and the text is included in the test object image, you might need to update the test object so that UFT can continue to identify it. You can modify the test object's image to include the updated text, or you can add a **similarity** identification property to the test object description, or lower the property's value, to enable UFT to match the test object with the object in the application despite the differences in the text.

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

Insight Test Object Properties

You can maintain the properties of Insight test objects in the same way as any other type of test object. However, keep in mind that for Insight test objects, UFT does not retrieve property values from the application. Therefore, you cannot add any properties from the run-time object to the description of the test object.

The main identification property of an Insight test object is its test object image. Additional properties that can help create a unique description of the object are:

- "Ordinal Identifiers" (described on page 1209)
- "Visual Relation Identifiers" (described on page 1209)
- Similarity (see "Insight Test Object Descriptions" on page 1173)

This section also includes:

- "Specifying or Modifying Property Values" below
- "Modifying an Insight test object's image" below
- "Updating Identification Properties from an Object in Your Application" on the next page
- "Restoring Default Mandatory Properties for a Test Object" on the next page
- "Renaming Test Objects " on the next page
- "Adding Properties to a Test Object Description" on page 1208
- "Defining New Identification Properties" on page 1208
- "Ordinal Identifiers" on page 1209

Specifying or Modifying Property Values

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

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

You can also find and replace specific identification property values. For details, see "Find and Replace Dialog Box (Object Repository)" on page 1245.

> **Note:** In some cases, the Smart Identification mechanism may enable UFT to identify a test object, even when some of its property values change. However, if you know about property value changes for a specific test object, you should try to correct the test object definition so that UFT can identify the test object from its basic object description. For details on the Smart Identification mechanism, see "Configuring Object Identification" on page 1310.

Modifying an Insight test object's image

An Insight test object's test object image is its main description property.

You can update an Insight object's image in the object repository by retaking a snapshot of the control in the application. Alternatively, you can use the snapshot previously stored with the test object to adjust the borders of the image used to identify the control.

Updating Identification Properties from an Object in Your Application

You can update a test object in your object repository by selecting the corresponding object in your application and relearning its properties and property values from the application. When you update a test object description in this way, all currently defined properties and values are overwritten, including description properties and values, the ordinal identifier, and Smart Identification information. An Insight test object's image is not updated.

The updated object description is based on the current definitions in the "Object Identification Dialog Box" (described on page 1325). Only the object-specific comments, if any, are retained.

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

You can also use this option to update an object that you defined (using the **Object > Define New Test Object** option) before the application was completely developed, and as a result some of the identification properties and values are missing in the test object description, or are no longer sufficient to identify the object. For details on the **Define New Test Object** option, see "Define New Test Object Dialog Box" on page 1240.

> **Note:** If you just want to restore the original test object description property set, while retaining any property values you have modified, you can use the **Restore mandatory property set** option. For details, see "Restoring Default Mandatory Properties for a Test Object" below.

Restoring Default Mandatory Properties for a Test Object

You can restore the default properties for a selected test object. When you restore the default properties, it restores the mandatory property set defined for the selected object class, based on the settings that were set in the "Object Identification Dialog Box" (described at 1325) at the time the object was learned. If you added or removed properties to or from the description, those changes are overwritten. However, if property values were defined or modified for any of the mandatory properties, those values are not modified when you choose this option. In addition, restoring the default mandatory property set does not change the values for the ordinal identifier or Smart Identification settings for the test object.

You can use the **Restore mandatory property set** option to restore the object description property set to the mandatory properties that were defined for that class when your object was learned. If the mandatory properties in the "Object Identification Dialog Box" are currently different for this test object class than they were when your object was learned, and you want to use the new definition, you can use the **Update From Application** option, which relearns the object properties and values based on the current definitions in the Object Identifications dialog box. For more details, see "Updating Identification Properties from an Object in Your Application" above.

Renaming Test Objects

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

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

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

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

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

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

Tip: If you do not want to automatically update test object names in the action or component for all occurrences of the test object, you can clear the **Automatically update test and component steps when you rename test objects** check box in the General pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> General** node). If you clear this option, you will need to manually change the test object names in all steps in which they are used, otherwise your run session will fail. For details, see "General Pane (Options Dialog Box > GUI Testing Tab)" on page 536.

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

Adding Properties to a Test Object Description

You can add to the list of properties that UFT uses to identify an object. For each object class, UFT has a default property set that it uses for the object description for a particular test object. You can use the "Add Properties Dialog Box" (described on page 1231) to change the properties that are included in the test object description.

Note: You can also add any valid identification property to a test object description, even if it does not appear in the Add Properties dialog box. For details, see "New Property Dialog Box" on page 1232.

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

You can also change the properties that identify an object if you want to reference objects using properties that UFT did not learn automatically when it learned the object. For example, suppose you are testing a Web site that contains an archive of newsletters. The archive page includes a hypertext link to the current newsletter and additional hypertext links to all past newsletters.

The text in the first hypertext link on the page changes as the current newsletter changes, but it always links to a page called current.html. Suppose you want to create a step in your test or component in which you always click the first hypertext link in your archive page. Since the news is always changing, the text in the hypertext link keeps changing. You need to modify how UFT identifies this hypertext link so that it can continue to find it.

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

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

Note: You can also modify the set of properties that UFT learns when it learns objects from a particular object class using the "Object Identification Dialog Box" (described on page 1325). Such a change generally affects only those objects that UFT learns after you make the change. For details, see "Configuring Object Identification" on page 1310. You can also apply the changes you make in the Object Identification dialog box to the descriptions of all objects in an existing test or component using the **Update Run Mode** option. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

Defining New Identification Properties

You can add any valid identification property to a test object description, even if it does not appear in the "Add Properties Dialog Box" (described on page 1231).

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

Ordinal Identifiers

An ordinal identifier assigns a numerical value to a test object that indicates its order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). This ordered value provides a backup mechanism that enables UFT to create a unique description to recognize an object when the defined properties are not sufficient to do so. For details on ordinal identifiers, see "Ordinal Identifiers" on page 1312.

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

Visual Relation Identifiers

Relevant for: GUI tests and components

When testing applications with multiple identical objects, UFT assigns an ordinal identifier to each test object. This may lead to unreliable object identification. However, it may not (immediately) result in a failed step. For details, see "Ordinal Identifiers" on page 1312.

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

You define visual relations in the "Visual Relation Identifier Dialog Box" (described on page 1251), which is accessible from the local or shared object repository, or from the "Object Properties Dialog Box" (described on page 1271).

How Visual Relation Identifiers Work

Suppose that someone shows you a photograph of identical twins sitting at different desks in a classroom and asks you to remember the differences between them so that you can identify each twin successfully when shown different photographs at a later time.

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

One way to remember which twin is which is by identifying twin A in this manner: Always look for the blue school bag and the twin's desk partner to locate twin A.

UFT uses visual relation identifiers in a similar manner. It compares the relative locations of the test objects you defined in the visual relation identifier with the multiple identical objects.

How UFT Uses Visual Relation Identifiers

- During a run session, UFT first attempts to identify the test object using the object's description properties. For details, see "How UFT Identifies Objects During a Run Session" on page 1169.

- If UFT finds one or more objects matching the test object's description, it attempts to identify the object using the visual relation identifier. For details, see "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 1225.

- After the visual relation identifier is applied, if no objects or more than one object is found, the visual relation identifier fails, and UFT continues to Smart Identification (when defined for that test object class). For details, see "Smart Identification" on page 1316.

- After Smart Identification is applied, if no objects or more than one object is found, no ordinal identifiers are used for that test object.

For general considerations on working with visual relation identifiers, see "Considerations for Working with Visual Relation Identifiers" on page 1256.

For a workflow of the general object identification process, see "Object Identification Process Workflow" on page 1188.

Tasks

How to Add a Test Object to an Object Repository

Relevant for: GUI tests and components

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

This section includes:

- "Add test objects to the object repository using the Add Objects to Local or Add Objects option" on the next page
- "Add an Insight test object to the object repository using the Add Insight Object or Add Insight Object to Local button" on the next page
- "Add a test object to the local object repository while adding a step to your test or component" on the next page
- "Define a new test object" on page 1213
- "Add a test object to the object repository using the Object Spy dialog box" on page 1213
- "Add a test object to the local object repository using the View/Add Object option from the Active Screen (tests only)" on page 1213
- "Add a test object to the local object repository by inserting a step from the Active Screen (tests only)" on page 1213

Note:

- You can add a test object to the local object repository only if that test object does not already exist in a shared object repository that is associated with the action or component. If a test object already exists in an associated shared object repository, you can add it to the local object repository using the **Copy to Local** option. For details, see "Local Copies of Objects from Shared Object Repositories " on page 1261.
- You cannot add **WinMenu** objects directly to an object repository using the **Add Objects to Local** button in the Object Repository window or the **Add Objects** button in the Object Repository Manager. If you want to add a **WinMenu** object to the object repository, you can use the **Add Objects** or **Add Objects to Local** button to add its parent object and then select to add the parent object together with its descendants, or you can record a step on a **WinMenu** object and then delete the recorded step.

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

1. Perform one of the following:

   - To add the test object to the *local* repository, making it available only in the current action or component:

     In the Object Repository window, Select **Object > Add Objects to Local** or click the **Add Objects to Local** toolbar button ➕ .

   - To add the test object to a *shared* repository, making it available for multiple actions or components:

     In the Object Repository Manager, select **Object > Add Objects** or click the **Add Objects** toolbar button ➕ .

     UFT and the Object Repository window or Object Repository Manager are hidden, and the pointer changes into a pointing hand. In some environments, as you move the pointing hand over your application, the test objects are highlighted. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.

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

3. If the location you click is associated with more than one object, the "Object Selection Dialog Box " (described on page 1195) opens. Select the object you want to add to the repository and click **OK**.

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

   If the object you select in the Object Selection dialog box is a parent (container) object, such as a browser or page in a Web environment, or a dialog box in a standard Windows application, the "Define Object Filter Dialog Box" (described on page 1242) box opens. The Define Object Filter dialog box retains the settings that you defined in the previous add object session.

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

Add an Insight test object to the object repository using the Add Insight Object or Add Insight Object to Local button

For details, see "How to Add an Insight Object to the Object Repository" on page 1214.

Add a test object to the local object repository while adding a step to your test or component

You can add a test object to the local object repository by choosing it from your application when you add or edit a step in your test or component.

You do this in the "Select Test Object Dialog Box" (described on page 958), which you can open when selecting an item for a step in the Keyword View or from the Step Generator.

Define a new test object

1. Select the object under which you want to define the new object, according to the correct object hierarchy.

2. Click the **Define New Test Object** button or select **Object > Define New Test Object**. The "Define New Test Object Dialog Box" (described on page 1240) opens.

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

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

2. Click the **Add Object** button . Depending on from where you opened the Object Spy dialog box, the object is added to the local or shared object repository. For details, see "Object Spy Dialog Box " on page 1189.

Add a test object to the local object repository using the View/Add Object option from the Active Screen (tests only)

1. If the Active Screen is not displayed, select **View > Active Screen**.

2. Select a step in your test whose Active Screen contains the object that you want to add to the object repository.

3. In the Active Screen, right-click the object you want to add and select **View/Add Object**.

4. If the location you clicked is associated with more than one object, the "Object Selection Dialog Box " on page 1195 opens. Select the object you want to add to the object repository, and click **OK** to close the Object Selection dialog box.

5. The "Object Properties Dialog Box" (described on page 1271) opens and displays the default identification properties for the object.

Add a test object to the local object repository by inserting a step from the Active Screen (tests only)

1. If the Active Screen is not displayed, select **View > Active Screen**.

2. Select a step in your test whose Active Screen contains the object for which you want to add a step.

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

4. If the location you clicked is associated with more than one object, the "Object Selection Dialog Box " (described on page 1195) opens. Select the object for which you want to add a step, and click **OK**.

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

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

How to Add an Insight Object to the Object Repository

Relevant for: GUI tests and components

This task describes how to add an Insight object to the object repository. You can add the object directly from the application, or even from a picture of the object displayed on your screen.

This task includes the following steps:

- "Select the Add Insight Object or Add Insight Object to Local command" below
- "Select the learning mode you want to use to add the Insight object - optional " below
- "In the application, select the control for which you want to add an Insight object" on page 1216
- "Results " on page 1216

1. **Select the Add Insight Object or Add Insight Object to Local command**

   Perform one of the following:

   - To add the test object to the *local* object repository, making it available only in the current action or component:

     Open the Object Repository window, select **Object > Add Insight Object to Local** or click the **Add Insight Object to Local** toolbar button .

   - To add the test object to a *shared* object repository, making it available for multiple actions and components:

     Open the Object Repository Manager, select **Object > Add Insight Object** or click the **Add Insight Object** toolbar button .

2. **Select the learning mode you want to use to add the Insight object - optional**

   - If the "Select Learn Mode Dialog Box" (described on page 1248) opens, click the button for the mode you want to use.

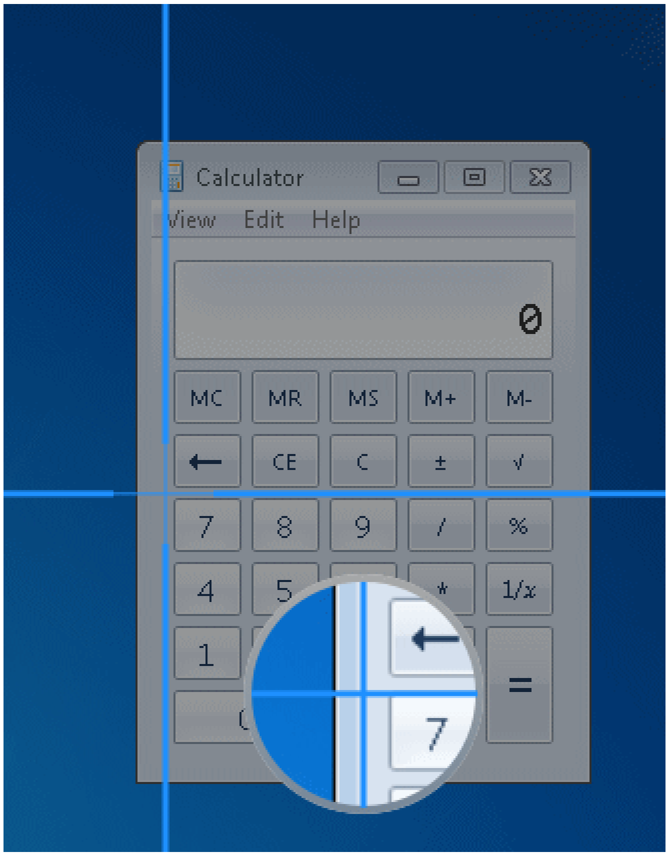
For details about the different modes, see "Adding Insight Test Objects" on page 1200.

- If you previously selected **Do not show me again** on the Select Learn Mode dialog box, it does not open, and the learning session begins using the mode you used most recently. (To instruct UFT to display the Select Learn Mode dialog box again, select **Tools > Options > GUI Testing** tab **> Insight** node and then select **Display Select Learn Mode dialog box**.)

  UFT and the Object Repository window or Object Repository Manager are hidden, and the pointer changes according to the learning mode, enabling you to select the control on your screen:

  - **Automatic mode.** The pointer changes to a pointing hand.

  - **Manual mode.** The pointer changes to a crosshair, with an adjacent circle displaying a magnified image of the area around the center of the crosshair.

    By holding the left CTRL, you can temporarily change the pointing hand or crosshair to a standard pointer, enabling you to change the window focus or perform operations in UFT or in your application.

For more details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.

3. **In the application, select the control for which you want to add an Insight object**

   a. Depending on the learning mode you are using, do one of the following:

      - **Automatic mode.** Click on the control.

      - **Manual Mode.** Use the crosshair to draw a rectangle around the control.

      UFT takes a snapshot of the control, automatically detecting the control's borders, or using the borders that you draw, and the Add Insight Test Object dialog box (described on page 1233) opens.

   b. In the Add Insight Test Object dialog box, you can:

      - Adjust the borders of the image saved with the test object in the object repository.

      - Take a new snapshot to replace the image entirely.

      - Specify areas to exclude from the test object image. UFT will ignore these areas when it searches for the image on the screen to identify the object.

      - Modify the test object's ClickPoint. This is the location to click in the control when running a test object method on it.

   c. Click **Save** (in the Change Test Object Image Dialog Box).

4. **Results**

   An InsightObject, named InsightObject, is added to the object repository, under the test object that represents the application or window that contains the control.

   - If an object by this name already exists under the same parent, the new object is named with an incremental suffix.

   - If the parent object does not already exist in the object repository, the parent is also added to the object repository.

   In the object repository you can do any of the following, if relevant, to improve the readability and efficiency of your test or component:

   - Rename the test object to a name that describes the control it represents. (Recommended)

   - Move the test object within the test object hierarchy: If you place it under another test object, then UFT searches for the object in the application only within its parent test object. If you move the Insight test object to be a top-level object, then UFT searches for the object anywhere on the screen.

For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" below.

- Add a **similarity** identification property to the test object description. For details about this property, see the **InsightObject Identification Properties** section in the *HP UFT Object Model Reference for GUI Testing*.

- Modify the ordinal identifier created for the test object. For details, see "Ordinal Identifiers" on page 1312.

- Define visual relation identifiers for the test object. For details, see "Visual Relation Identifiers" on page 1209.

> **Tips:**
>
> - When working in the Editor, you can open the object repository by double-clicking an InsightObject's image. The object repository opens with the InsightObject selected. (If test object images are not displayed in the Editor, you can display them using the relevant option in the **Tools > Options > GUI Testing** tab > **Insight** pane.)
>
> - When working in the object repository, you can use the **Highlight in Application** option to see which controls UFT identifies using the current test object's description.

When you have finished modifying all of the Insight test objects to your satisfaction in the object repository, you can delete all of the snapshots to reduce the amount of disk space used (in the Object Repository window or the Object Repository Manager, **Tools > Delete Insight Snapshots**). This does not delete the test object images used for object identification.

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

Relevant for: GUI tests and components

The following steps describe how to copy, paste, move, and delete objects in an object repository. This functionality is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories:

- "Move an object to a different location within an object repository" on the next page
- "Copy an object to a different location within an object repository" on the next page
- "Move or copy an object without its child objects" on the next page
- "Cut, copy, and paste objects within an object repository" on the next page
- "Cut, copy, and paste objects between shared object repositories" on the next page

- "Copy objects from one shared object repository to another" below
- "Move objects from one shared object repository to another" below
- "Delete an object from the object repository" on the next page
- "Add test objects from a local or shared object repository to your test or component " on the next page

Move an object to a different location within an object repository

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

Copy an object to a different location within an object repository

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

Move or copy an object without its child objects

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

Cut, copy, and paste objects within an object repository

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

Cut, copy, and paste objects between shared object repositories

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

Copy objects from one shared object repository to another

1. In the Object Repository Manager, open the required shared object repositories and select **Window > Cascade**. All open shared object repositories are displayed as separate windows.
2. Drag the object from one window and drop it at the required location in the other window.

Move objects from one shared object repository to another

1. In the Object Repository Manager, open the required shared object repositories and select **Window > Cascade**.
2. Press the CTRL key while you drag the object from one window and drop it at the required location in the other window. Note that moving an object removes it from one shared object

repository and adds it to the other shared object repository.

You can also copy objects from a shared object repository to the local object repository to modify them locally. For more details, see "Local Copies of Objects from Shared Object Repositories " on page 1261.

Delete an object from the object repository

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

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

You can drag and drop test objects from a shared or local object repository to your test or component. When you drag and drop a test object to your test or component, UFT inserts a step with the default operation for that test object in your test or component.

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

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

For details on adding objects to your test or component, see:

- "Toolbox Pane Overview" on page 506
- "Object Repository Window" on page 1274

How to Locate an Object in an Object Repository

Relevant for: GUI tests and components

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

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

Find an Object in an Object Repository

1. Make sure that the relevant object repository is open (in the Object Repository window or Object Repository Manager).
2. Click the **Find & Replace** button 🔍. The "Find and Replace Dialog Box (Object Repository)

" (described on page 1245) opens.

Highlight an Object in Your Application

1. Make sure your application is open to the correct window or page.

   > **Tip:** If you want to highlight an Insight test object located on the desktop (and not in an application), make sure that UFT is not hiding part of the object.

2. Select the test object you want to highlight in your object repository.

3. Click the **Highlight in Application** button.

Locate an Object from Your Application in the Object Repository

1. Make sure your application is open to the correct window or page.

2. Click the **Locate in Repository** button.

   UFT is hidden, and the pointer changes into a pointing hand. In some environments, as you move the pointing hand over your application, the test objects are highlighted.

3. Use the pointing hand to click the required object in your application. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.

- If the location you clicked is associated with more than one object, the "Object Selection Dialog Box " (described on page 1195) opens. The selected object is highlighted in the object repository.

  > **Note:**
  >
  > - If the relevant object repository is not open or the object cannot be found, the object is not highlighted.
  > - In the Object Repository Manager, if more than one shared object repository is open, and UFT cannot locate the selected object in the active object repository, you can choose whether to look for the object in all of the currently open object repositories.
  > - **Locate in Repository** is not supported for Insight test objects.

How to Maintain Test Objects in Object Repositories

Relevant for: GUI tests and components

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

- "Specify an identification property value" below
- "Modify an Insight test object's image" on the next page
- "Update identification properties from an object in your application" on the next page
- "Restore the mandatory property set" on the next page
- "Rename test objects" on page 1223
- "Add properties to a test object description" on page 1223
- "Define a new identification property" on page 1223
- "Remove properties from a test object description" on page 1224
- "Specify an ordinal identifier" on page 1224
- "Define related objects for a specific test object" on page 1224

Specify an identification property value

1. In the Object Repository window or the Object Repository Manager, select the test object whose property value you want to specify.

   Tip:

   - For a test object in the local object repository, you can also right-click the step containing the test object and select **Object Properties**, and then make the following property value changes in the "Object Properties Dialog Box" (described on page 1271).
   - If you want to view all objects in the action or component, click the **View in Repository** button. The Object Repository window opens and displays all objects stored in the repository in a repository tree.
   - You can also open the object repository for the selected action or component by choosing **Tools > Object Repository Window** or by clicking the **Object Repository** toolbar button .

2. In the **Test object details** area, click in the value cell for the required property.

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

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

For more details, see "How to Define Values for Your Step Arguments " on page 928.

If you specified a constant value, it is shown in the **Value** column of the **Test object details** area. If you parameterized the value, the parameter name is shown with one of the following icons in the **Value** column. For details, see "Object Repository Window" on page 1274.

Modify an Insight test object's image

1. In the Object Repository window or Manager, select the test object whose image you want to modify.

   > **Tip:** From the Editor, you can open the Object Repository window with the Insight object selected, by double-clicking the test object's image in a step. (If test object images are not displayed in the Editor, you can display them using the relevant option in the **Tools > Options >** s **Testing** tab > **Insight** pane.)

2. In the **Test object image** area, click the **Change Test Object Image** button. For details, see "Change Test Object Image / Add Insight Test Object Dialog Box" on page 1233.

Update identification properties from an object in your application

1. In the object repository tree, select the test object whose description you want to update. (You cannot use this process to update the test object image of an Insight test object.)

2. Select **Object > Update from Application** or click the **Update from Application** button . UFT is hidden, and the pointer changes into a pointing hand. In some environments, as you move the pointing hand over your application, the test objects are highlighted. For more details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.

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

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

Restore the mandatory property set

1. In the object repository tree, select the test object whose description you want to restore.

2. In the **Test object details** area, click the **Restore mandatory property set** button .

3. Click **Yes** to confirm the operation. The test object's description properties are restored to the

mandatory property set for the selected object class at the time that the object was learned.

Rename test objects

1. In the object repository tree of the Object Repository window or Manager, select the test object that you want to rename.

2. In the **Name** box in the Object Properties pane, enter the new name for the test object. Then click anywhere else to remove the focus from the object. For a list of naming conventions, see the **Test object name** section in "Troubleshooting - Naming Conventions" on page 2269. Test object names are not case-sensitive.

Add properties to a test object description

1. In the object repository tree of the Object Repository window or Manager, select the test object whose description you want to modify.

2. In the **Test object details** area, click the **Add description properties** button +.

3. The "Add Properties Dialog Box" (described on page 1231) opens listing the properties that can be used to identify the object (properties that are not already part of the test object description).

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

Define a new identification property

1. In the object repository tree of the Object Repository window or Manager, select the test object for which you want to define a new property.

2. In the **Test object details** area, click the **Add description properties** button +. The "Add Properties Dialog Box" (described on page 1231) opens.

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

3. Click the **Define new property** button ✳. The "New Property Dialog Box" (described on page 1232) opens.

Remove properties from a test object description

1. In the object repository tree of the Object Repository window or Manager, select the test object whose description you want to modify.

2. In the **Test object details** area, select one or more properties that you want to remove from the test object description.

   > **Tip:** For an object in the local object repository, you can also select the required test object, right-click and select **Object Properties**, and then perform the following steps in the "Object Properties Dialog Box" (described on page 1271).

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

Specify an ordinal identifier

1. In the object repository tree of the Object Repository window or Manager, select the test object whose ordinal identifier you want to specify.

2. In the **Test object details** area, click in the cell to the right of the **Type, Value** cell under the **Ordinal identifier** row.

   > **Tip:** For an object in the local object repository, you can also select the required test object, right-click and select **Object Properties**, click in the cell to the right of the **Type, Value** cell under the **Ordinal identifier** row, and then perform the following steps in the "Object Properties Dialog Box" (described on page 1271).

3. Click the **Browse** button. The "Ordinal Identifier Dialog Box" (described on page 1247) opens.

Define related objects for a specific test object

1. In the **Visual Relation Identifier Settings** row of the Object Repository window or Object Properties dialog box, click in the **Value** cell.

2. Click the **Browse** button in the cell. The "Visual Relation Identifier Dialog Box" (described on page 1251) opens.

3. Set the options for the visual relation identifier. For details, see "Visual Relation Identifier Dialog Box" on page 1251.

Results:

- The visual relation identifier is added to the selected test object, and the text in the **Value** cell indicates that a visual relation identifier is defined.

- Any related objects you specified are linked to the test object for which you are using a visual relation identifier. You cannot define visual relations for those objects.

- The **Ordinal identifier** property is disabled in the **Object Details** area of the local or shared object repository, and is not used during the object identification process. However, UFT still uses this property during the learn process, when comparing existing objects with the objects to be learned, and therefore the ordinal identifier value should not be manually changed or removed.

For considerations on working with visual relation identifiers, see "Considerations for Working with Visual Relation Identifiers" on page 1256.

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

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

Relevant for: GUI tests and components

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

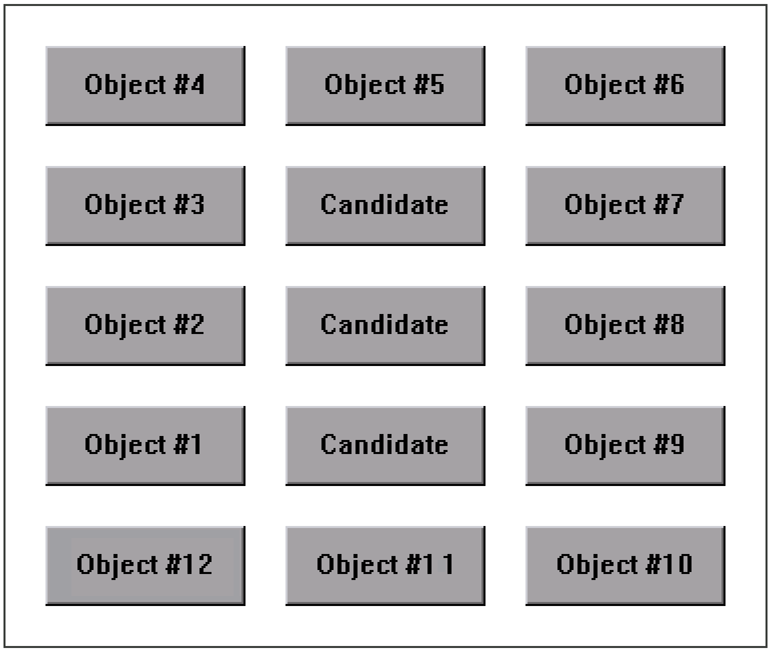
> **Note:** For a task related to this scenario, see "How to Maintain Test Objects in Object Repositories" on page 1220.

This scenario includes the following steps:

- "Background" below
- "Access the Visual Relation Identifier dialog box" on the next page
- "Highlight the objects in the application that match the test object's description" on the next page
- "Define the first related test object using horizontal visual relations" on page 1227
- "Define the second related object using vertical visual relations" on page 1228
- "Define the third related object using distance visual relations" on page 1229
- "Results" on page 1230

1. Background

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



- When UFT learned the objects in the application, it assigned an ordinal identifier to each Candidate test object.

- For the purpose of this exercise, Object #1 and Object #9 make up an object pair, which is always to the left and right of the Candidate object to identify. You want to instruct UFT to identify the instance of the Candidate object that is located between the Object #1 and Object #9 object pair during every run session, even if the sorting order of the object pairs changes between run sessions.

2. **Access the Visual Relation Identifier dialog box**

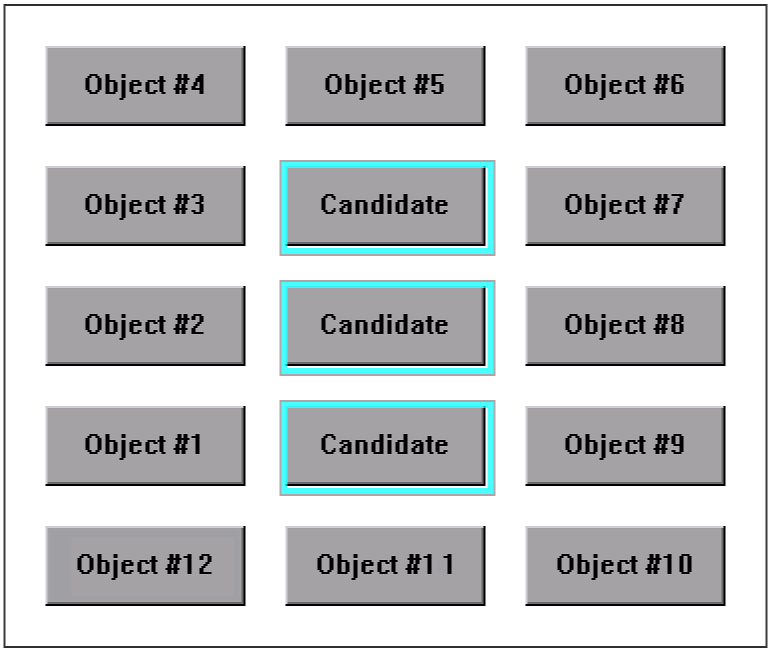
   a. In UFT, open the relevant object repository and select the Candidate test object to identify.

   b. Verify that you have selected the correct test object by selecting **View > Highlight in Application**, and making sure that the correct object is highlighted in the application.

   c. Open the "Visual Relation Identifier Dialog Box" (described on page 1251).

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

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



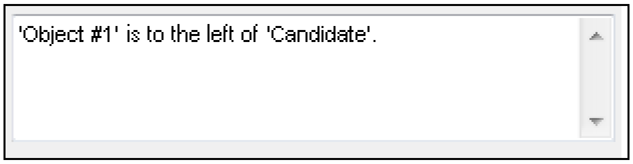
You then click the **Preview** button again to restore the UFT window.

4. **Define the first related test object using horizontal visual relations**

   a. In the **Related Objects** area, click the **Add** button. The "Select Test Object Dialog Box" (described on page 958) opens, enabling you to either select a test object from the object repository, or add an object from the application.

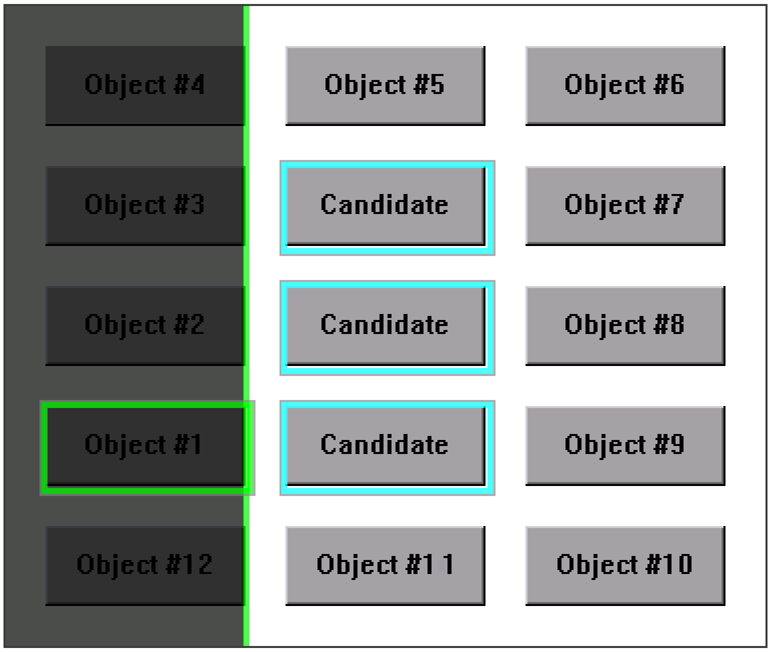
      For the purpose of this scenario, the first related test object is Object #1, which is located to the left of the Candidate object in the application.

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



'Object #1' is to the left of 'Candidate'.

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

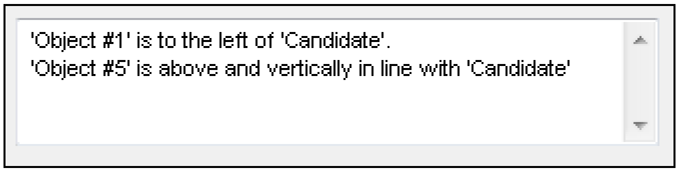


5. Define the second related object using vertical visual relations

a. In the **Related Objects** area, click the **Add** button. The "Select Test Object Dialog Box" (described on page 958) opens, enabling you to select or add another object.

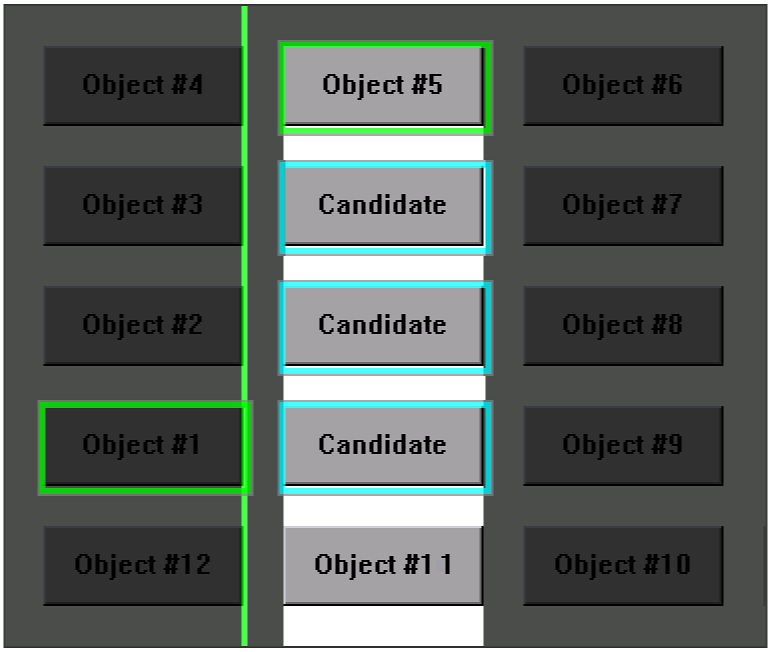
For the purpose of this scenario, the second related test object is Object #5, which is located above and vertically in line with the Candidate object.

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



'Object #1' is to the left of 'Candidate'.
'Object #5' is above and vertically in line with 'Candidate'

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

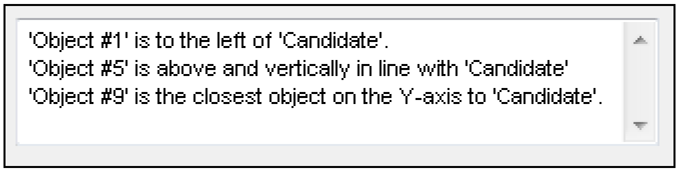


6. **Define the third related object using distance visual relations**

a. In the **Related Objects** area, click the **Add** button. The "Select Test Object Dialog Box" (described on page 958) opens, enabling you to select another test object.

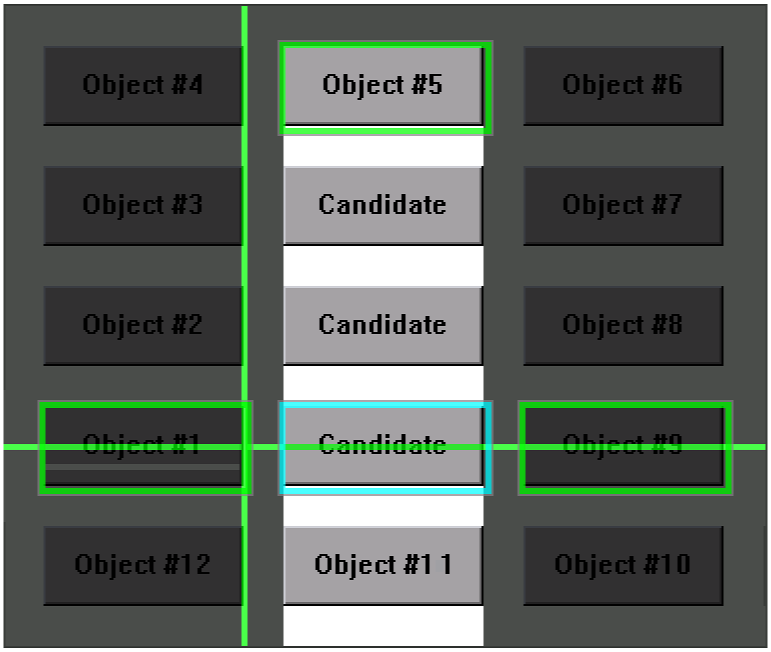
For the purpose of this scenario, the third related test object is Object #9, which is the closest object to the right of the Candidate object.

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



'Object #1' is to the left of 'Candidate'.
'Object #5' is above and vertically in line with 'Candidate'
'Object #9' is the closest object on the Y-axis to 'Candidate'.

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



7. **Results**

After you finish defining all of the necessary visual relations:

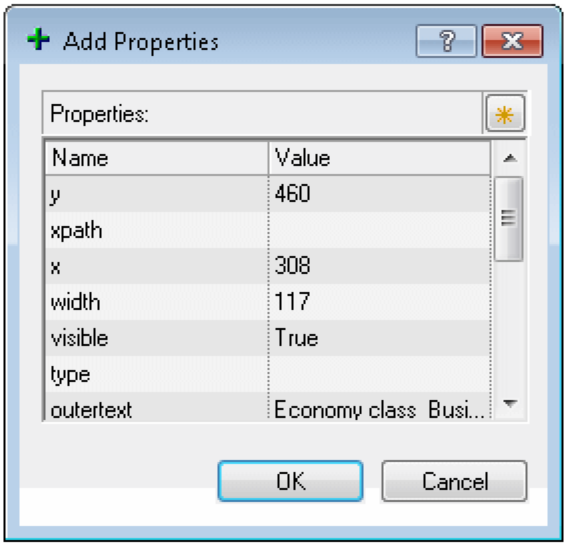
- The desired Candidate object is the only object in the application that is identified when you use **Preview**.
- UFT can now correctly identify the desired Candidate object during every run session, even if the user interface changes, as long as the Candidate object maintains it's relative location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

Reference

Add Properties Dialog Box

Relevant for: GUI tests and components

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



Name	Value
y	460
xpath	
x	308
width	117
visible	True
type	
outertext	Economy class Busi...

To access	From the "Object Repository Manager Main Window" (described on page 1297) or "Object Repository Window" (described on page 1274), in the **Test object details** area, click the **Add description properties** button + .
Important information	• Values for all properties are displayed only if the application that contains the object is currently open. If the application is closed, only values for properties that were part of the object description when the object was learned are shown. • You can resize the Add Properties dialog box to enable you to view long property values. • After you add a new property to the object description, you can modify its value. For details on modifying object property values, see "Specifying or Modifying Property Values" on page 1205.
Relevant tasks	"How to Maintain Test Objects in Object Repositories" on page 1220
See also	"Maintaining Identification Properties" on page 1204

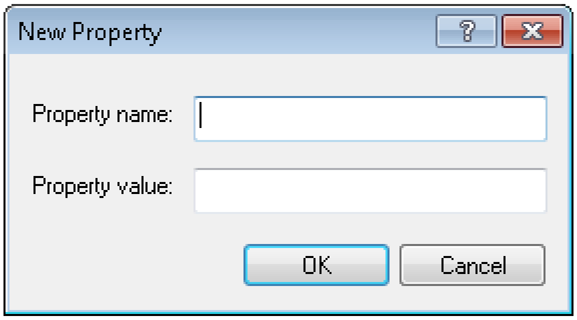
User interface elements are described below:

UI Elements	Description
✳	**Define new property.** Opens the "New Property Dialog Box" (described on page 1232).
<Properties list>	Select one or more properties to add to the test object description and click **OK**. You can also double-click a property to add it to the test object description. You can type the first letters of a property to highlight the first property in the list that matches the pattern.

New Property Dialog Box

Relevant for: GUI tests and components

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



To access	Click the **Define new property** button ✳ in the "Add Properties Dialog Box" (described on page 1231).
Important information	You must enter a valid identification property in the Property value edit box. If you enter an invalid property and then select it to be part of the object description, your run session will fail.
Relevant tasks	"How to Maintain Test Objects in Object Repositories" on page 1220

User interface elements are described below:

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

Change Test Object Image / Add Insight Test Object Dialog Box

Relevant for: GUI tests and components

UFT uses the image stored with an Insight test object to identify the relevant control in the application.

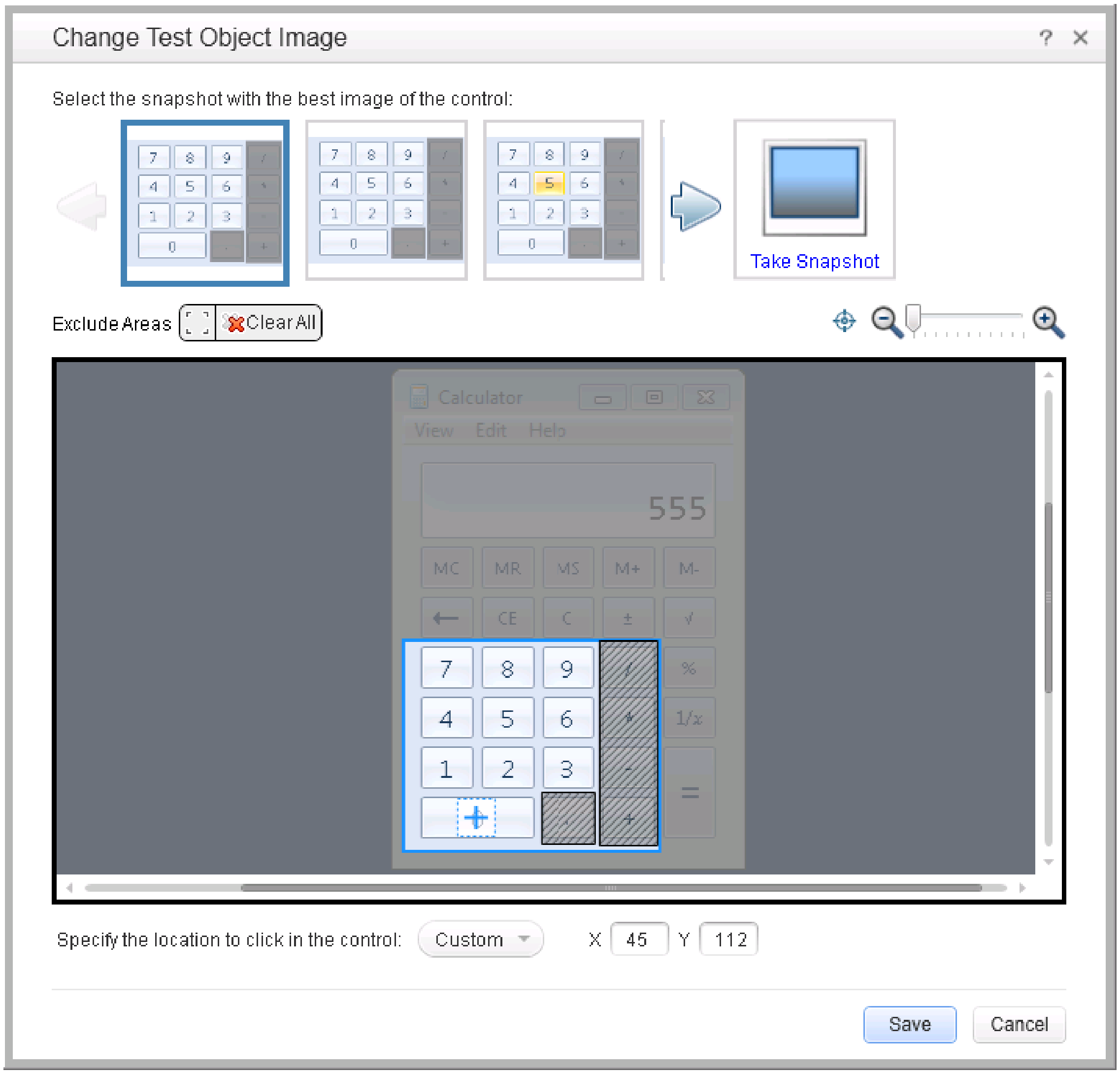
This dialog box enables you to:

- Adjust the borders of the image saved with the test object in the object repository.
- Take a new snapshot to replace the image entirely.
- Specify areas to exclude from the test object image. UFT will ignore these areas when it searches for the image on the screen to identify the object.
- Modify the test object's ClickPoint. This is the location to click in the control when running a test object method on it.

You can do this when editing an existing test object, or when adding a new one.

The image below shows the dialog box that opens when you click the **Change Test Object Image** button in the Object Repository for an Insight test object that was added to the object repository during a recording session.

The dialog box options differ slightly when opened in other circumstances.



Change Test Object Image

Select the snapshot with the best image of the control:

Exclude Areas

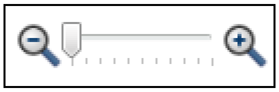
Specify the location to click in the control: Custom X 45 Y 112

To access	Do one of the following: **To change the image for an existing test object:** 1. Select an Insight test object in the Object Repository window or Object Repository Manager. 2. In the **Test object image** area, click the **Change Test Object Image** button. **To fine-tune the image during the process of adding a new test object:** Perform the first steps described in "How to Add an Insight Object to the Object Repository" on page 1214. This dialog box opens after you select the control in the application and a snapshot of it is created (automatically or manually).
Important Information	The insight mechanism is not sensitive to all differences in color, and does not support text recognition. Therefore, define the borders of your test object image around visually distinct borders, templates, shades, icons, or shapes. This is particularly relevant when capturing images that contain sharp contrasts.
Relevant tasks	• "How to Add an Insight Object to the Object Repository" on page 1214 • "How to Record a Test or Component Using Insight Recording" on page 860
See also	• "Identifying Objects Using Insight" on page 1176 • "Select Learn Mode Dialog Box" on page 1248

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

UI Elements	Description
<snapshot sequence>	A series of images of the control, taken during a recording session. The images are derived from snapshots taken of the entire application. • **The highlighted image** is the one stored with the test object. • **The sequence** may show additional images, taken shortly before and/or after the stored one. **Note:** • When you select an image, the **snapshot display** below displays the corresponding application snapshot. You can then adjust the image stored with the test object by moving or resizing the highlighted rectangle in the snapshot display. • You can configure the size and number of snapshots to include in the sequence (Select **Tools > Options > GUI Testing** tab > **Insight** node). • The snapshot sequence is available only when editing a test object that was created during a recording session.
⇦ ⇨	**Next/Previous.** Enable you to scroll through the images in the snapshot sequence and select another image to use. These buttons appear only when you have more images in the snapshot sequence than can fit in the dialog box.

UI Elements	Description
Take/Replace Snapshot	Enables you to manually take a snapshot of the control within the application. (This replaces the snapshot associated with this button, but not any snapshots in the snapshot sequence.) When you click this button, UFT is minimized, and the mouse pointer changes to a crosshair, which you can use to draw a rectangle around the control and take the snapshot. To help you accurately draw the borders of the control, a magnified view of the mouse location on the screen is displayed in a circle near the crosshair. By holding the left **CTRL**, you can temporarily change the crosshair to a standard pointer, enabling you to change the window focus or perform operations in UFT or in your application. **Note:** • Available only when editing an existing test object. • The button displays the image of the control, as defined by the rectangle highlighted within the snapshot associated with the button.
Exclude Areas toolbar	Enables you to specify areas to exclude from the test object image or clear all excluded areas. • **Select area to exclude/Select additional areas to exclude** ⬚. When you click this button, the mouse pointer changes to a crosshair, enabling you to draw a rectangle within the test object image. This does not clear existing excluded areas. To draw multiple areas to exclude, press and hold the **CTRL** key while you draw. • **Clear All.** Clears all excluded areas, so that the full test object image is included. To clear only one excluded area, hover over the area and press the **DELETE** key or click the **Clear area** ☒ button that pops up beside the area. **Note:** Make sure that the included area contains enough significant content to enable object recognition.

UI Elements	Description
⌖	**Bring the highlighted area into view.** Scrolls the snapshot of the application so that the highlighted rectangle is visible in the center of the snapshot display.
	**Zoom Out / Zoom In slider.** Enables you to zoom into the application snapshot to view it at a larger size (up to 200%), and to zoom back out, up to the original size of the snapshot. • Zoom in to see the borders of the control more clearly and adjust the image used for the control. • Zoom back out to see more of the context of the application in which the control is located.
<snapshot display>	Displays a snapshot of the application in which the control is located. This area also contains: • **A highlighted rectangle.** Defines the borders of the test object image stored with in the object repository. When you hover over this rectangle, it becomes editable and you can drag or resize it to accurately represent the control. If you resize this rectangle after selecting areas to exclude, then any parts of excluded areas that are located outside the new rectangle are cleared. • **A crosshair.** Specifies the location to click in the control. If you select **Custom** in **Specify the location to click in the control** below, you can modify the location of the crosshair by dragging it, or by editing its coordinates in the **X** and **Y** boxes. • **Gray rectangles within the test object image** (optional)**.** Excluded areas within the test object image that UFT ignores when it searches for the image on the screen to identify the test object. When you hover over an excluded area, it becomes editable and you can drag or resize it. If you press the **DELETE** key or click the **Clear area** ☒ button that pops up beside the area, the rectangle is removed, and the area is included in the test object image. To specify additional excluded areas, or to clear all of the areas at once, use the **Exclude Areas** toolbar.

UI Elements	Description
Specify the location to click in the control	The default location to click in the control when performing a test object method on the control during a run session. (This is also referred to as the object's ClickPoint.) If necessary, you can specify a different location when running a specific test object method by specifying coordinates in the relevant step. Possible values: • **Center.** Click in the center of the control. • **Custom.** Click the control at the specified location. If you select this option, the **X** and **Y** fields become available. **Example:** You may want to specify a **Custom** location for controls such as combo boxes, where clicking in the center would enable the control for typing, and clicking on the side opens the drop-down list. **Note:** If you want the ClickPoint for Insight objects set to **Center** when they are added during a recording session, clear the **Save the clicked coordinates as the test object's ClickPoint** option in the Insight pane of the Options dialog box (**Tools > Options > GUI Testing > Insight**). For details, see "Insight Pane (Options Dialog Box > GUI Testing Tab)" on page 561. For Insight objects added to the object repository using the **Add Insight Object** or **Add Insight Object to Local** button, the ClickPoint is initially set to **Center**.

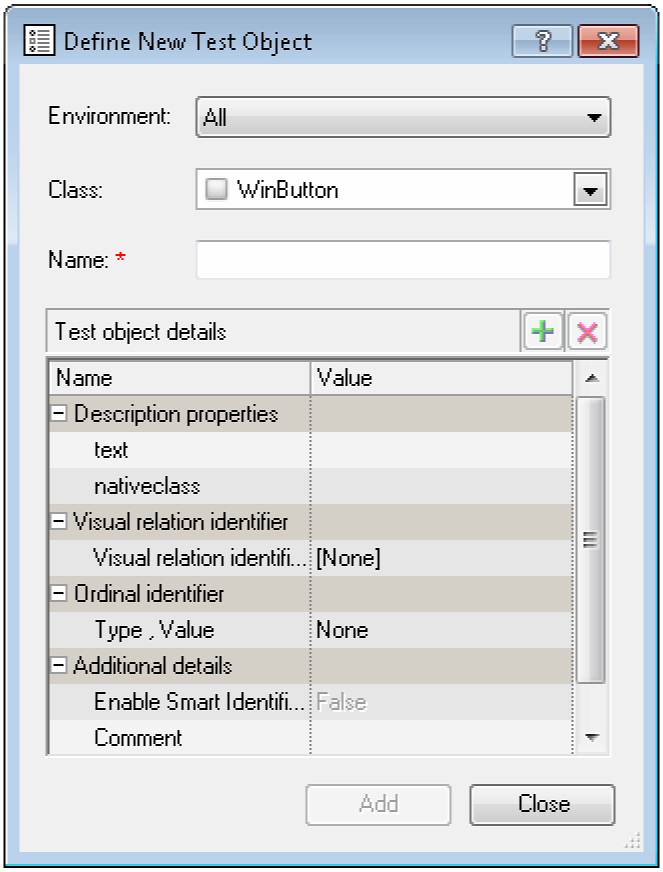
UI Elements	Description
X, Y	The x or y coordinate of the location to click in the control. (Relative to the top left corner of the highlighted triangle). **Note:** - Available when the location to click in the control is set to **Custom**. - You can modify these values by typing in the **X** or **Y** box or by dragging the crosshairs displayed within the highlighted rectangle. - You can specify a location to click that it outside the borders of the test object image. For example, the test object image used to identify an edit box or drop down list might be the text adjacent to the control, but you want the test object operations (Type, Click) to be carried out on the control itself.

Define New Test Object Dialog Box

Relevant for: GUI tests and components

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

.



Environment: All

Class: WinButton

Name: *

Name	Value
Description properties	
text	
nativeclass	
Visual relation identifier	
Visual relation identifi...	[None]
Ordinal identifier	
Type , Value	None
Additional details	
Enable Smart Identifi...	False
Comment	

To access	From the "Object Repository Manager Main Window" (described on page 1297) or the "Object Repository Window" (described on page 1274), select **Object > Define New Test Object**.
Important information	- You cannot use this dialog box to define Insight test objects. - When defining a new test object for windowless controls, you must create a new property named **acc_name**, and include it in the object description. **Note:** The **acc_name** property is not available in the Object Identification dialog box.
Relevant tasks	"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199
See also	"Defining New Test Objects" on page 1201

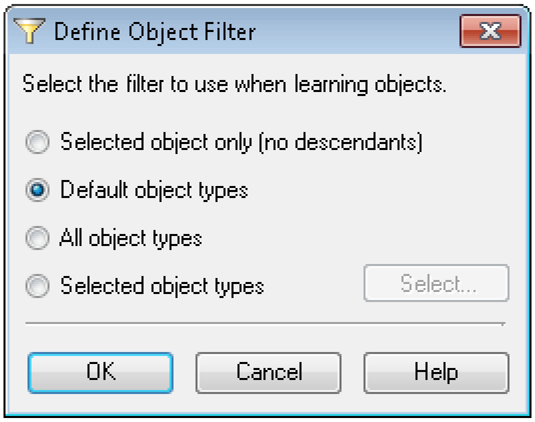
User interface elements are described below:

UI Elements	Description
Environment	The list of available environments. The test object classes associated with the selected environment are displayed in the **Class** box. **Note:** - The environments included in the **Environment** list correspond to the loaded add-ins. For details on loading add-ins, see the section on loading UFT add-ins in the *HP Unified Functional Testing Add-ins Guide*. - The **Environment** list might also include additional environments for which you or a third party developed support using UFTadd-in extensibility add-in extensibility.
Class	Select the class of the test object you want to define.
Name	Enter a name for the new test object. After you enter a name, the **Test object details** area is enabled.
Test object details	Define the properties and values for your test object. The Test object details area automatically contains the mandatory properties defined for the object class in the "Object Identification Dialog Box" (described on page 1325). You can add or remove properties as required, and define values for the properties. For details, see "Maintaining Identification Properties" on page 1204.

Define Object Filter Dialog Box

Relevant for: GUI tests and components

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


Define Object Filter

Select the filter to use when learning objects.

○ Selected object only (no descendants)

● Default object types

○ All object types

○ Selected object types

To access	Do one of the following: • In the Object Repository Manager window, select **Object > Navigate and Learn**. • In the Object Repository window, add a test object to the object repository using the **Add Objects to Local** and selecting an object that contains child objects. • In the Object Repository Manager, add a test object to the object repository using the **Add Objects** and selecting an object that contains child objects. For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199.
Relevant tasks	"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199

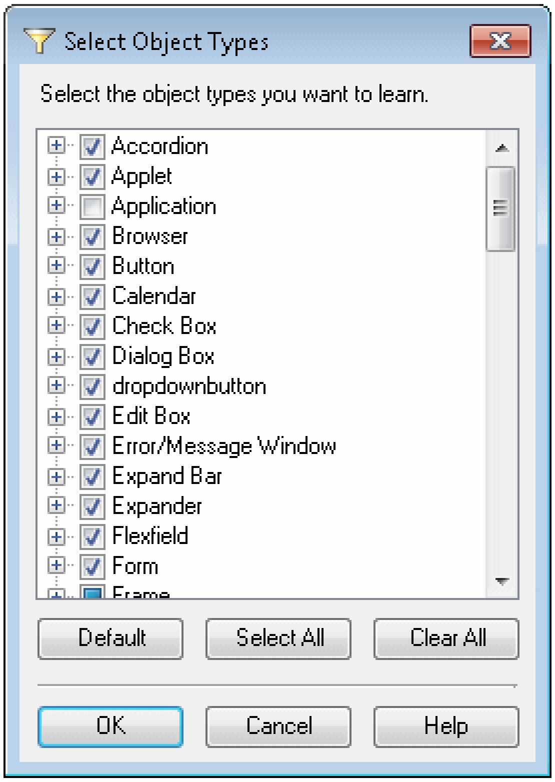
User interface elements are described below:

UI Elements	Description
Selected object only (no descendants)	Adds to the object repository the previously selected object's properties and values, without its descendant objects.
Default object types	Adds to the object repository the previously selected object's properties and values, with the properties and values of its descendant objects according to the object types specified by the default filter. You can see which objects are in the default filter by selecting **Selected object types**, clicking the **Select** button, and then clicking the **Default** button.
All object types	Adds to the object repository the previously selected object's properties and values, together with the properties and values of all of its descendant objects.
Selected object types	Adds to the object repository the previously selected object's properties and values, as well as the properties and values of its descendant objects according to the object types and classes you specify in the object filter. You specify the objects and classes in the filter by clicking the **Select** button and selecting the required items in the "Select Object Types Dialog Box" (described on page 1244).

Select Object Types Dialog Box

Relevant for: GUI tests and components

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



Select Object Types

Select the object types you want to learn.

- ☑ Accordion
- ☑ Applet
- □ Application
- ☑ Browser
- ☑ Button
- ☑ Calendar
- ☑ Check Box
- ☑ Dialog Box
- ☑ dropdownbutton
- ☑ Edit Box
- ☑ Error/Message Window
- ☑ Expand Bar
- ☑ Expander
- ☑ Flexfield
- ☑ Form
- Frame

To access	Click the **Select** button in the "Define Object Filter Dialog Box" (described on page 1242).

Important information	• When you define an object filter, it is automatically saved for future add object operations (performed from both the **Navigate and Learn** option and the **Add Objects** option). • The object types in this list are a generic grouping of objects according to the general object characteristics. For example, the List type contains list and list view objects, as well as combo boxes; the Table type contains both tables and grids. • The list shows all objects supported by the installed add-ins and is not specific to the object you selected. For some add-ins, certain child objects may be automatically filtered out and not added to the object repository when you choose to add all descendants of a specific object, even if those object types are selected in the list. If you want to add an object that is automatically filtered out, you can add it by selecting it in the "Object Selection Dialog Box " (described on page 1195). To check whether your add-in automatically filters out certain objects, see the *HP Unified Functional Testing Add-ins Guide*.
Relevant tasks	"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199
See also	"Adding Test Objects to the Local Object Repository from the Active Screen (GUI actions and scripted components only)" on page 1200

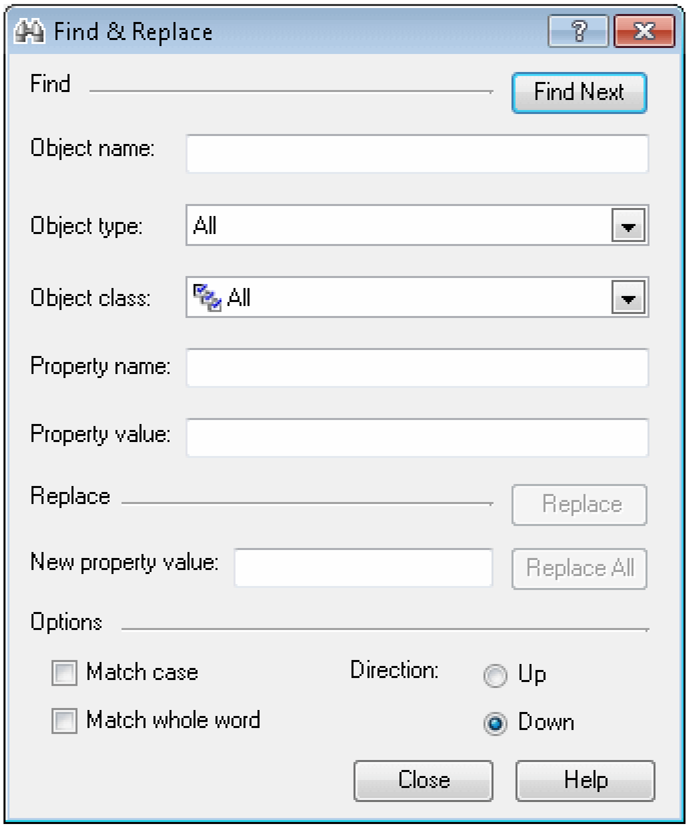
User interface elements are described below:

UI Elements	Description
Default	Restores the check box selections to their preset defaults. The preset defaults are equivalent to choosing the **Default object types** option in the "Define Object Filter Dialog Box" on page 1242
Select All	Selects all the check boxes.
Clear All	Clears all the check boxes.

Find and Replace Dialog Box (Object Repository)

Relevant for: GUI tests and components

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



Object name:

Object type: All

Object class: All

Property name:

Property value:

New property value:

□ Match case

□ Match whole word

Direction: ○ Up ● Down

To access	Click the **Find & Replace** button in the "Object Repository Manager Main Window" on page 1297.
Important information	- The functionality in this dialog box is available in the Object Repository window for the local object repository, and the Object Repository Manager for shared object repositories. - You can search using any or all of the criteria in the Find and Replace dialog box. - The Find and Replace dialog box can find checkpoint and output values only by searching for the object name. - You cannot use the Find and Replace dialog box to replace property or object names. You cannot replace property values in a read-only test or component.
Relevant tasks	"How to Locate an Object in an Object Repository" on page 1219
See also	"Locating Objects" on page 1202

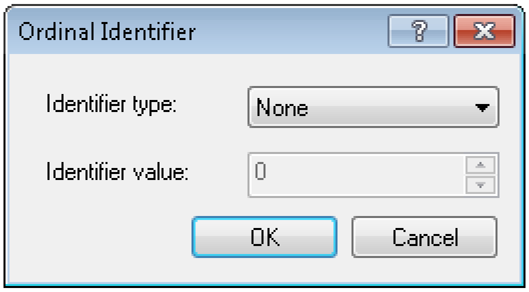
User interface elements are described below:

UI Elements	Description
Object name	The name or partial name of the object you want to find.
Object type	The type of object you want to find, for example, Button. The object types in this list are a generic grouping of objects according to the general object characteristics. For example, the **List** type contains list and list view objects, as well as combo boxes; the **Table** type contains both tables and grids.
Object class	The class of object you want to find, for example, WebButton. The classes available depend on the selection you made in the **Object type** box.
Property name	The name or partial name of the property you want to find.
Property value	The property value or partial property value you want to find.
New property value	The new property value if you specified a property value and want to replace it with a different value.
Options	• **Match case**. Select this option if you want the search to distinguish between upper and lower case letters. • **Match whole word.** Select this option if you want the search to find only complete words that exactly match the single word you entered. • **Direction.** The direction you want to search.

Ordinal Identifier Dialog Box

Relevant for: GUI tests and components

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



Identifier type: None

Identifier value: 0

To access	1. From the "Object Repository Manager Main Window" (described on page 1297) or "Object Repository Window" (described on page 1274), in the **Test object details** area, click in the **Value** column of the **Type, Value** row in the **Ordinal identifier** section. 2. Click the **Browse** button in that cell.
Relevant tasks	"How to Maintain Test Objects in Object Repositories" on page 1220
See also	"Maintaining Identification Properties" on page 1204

User interface elements are described below:

UI Elements	Description
Identifier type	- **Index.** The order in which the object appears within the parent window, frame, or dialog box relative to other objects with an otherwise identical description. - **Creation Time.** (Browser objects only). The order in which the browser was opened relative to other open browsers with an otherwise identical description. This identifier type is only available if more than one Browser object was open when the test object was learned. - **None.** Does not specify an ordinal identifier. This is the default value if UFT did not learn an ordinal identifier.
Identifier value	The numeric value of the ordinal identifier.

Select Learn Mode Dialog Box

Relevant for: GUI tests and components

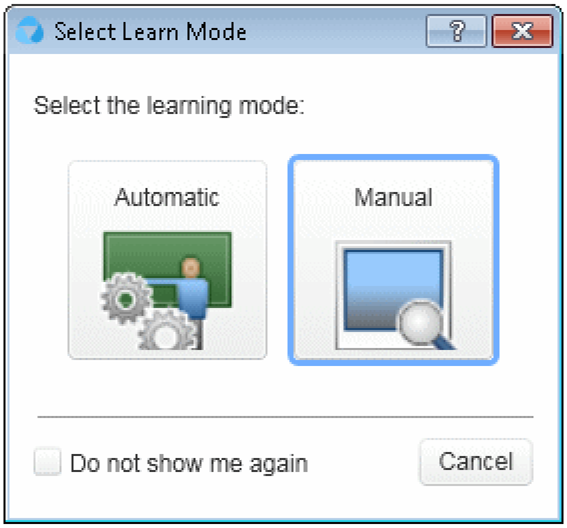
This dialog box enables you to specify which learning mode to use when adding an Insight test object to the object repository.

- When using the automatic mode, you click on the control in the application and UFT automatically determines the borders of the control.

- When using the manual mode, you draw a rectangle around the control in the application, specifying the borders of the control.

In both cases, UFT takes a snapshot of the control within the application and the Add Insight Test Object dialog box (described on page 1233) opens, enabling you to adjust the image that will be saved with the new test object.

.



Select the learning mode:

To access	Do one of the following: • In the "Object Repository Window" (described on page 1274), click **Add Insight Object to Local** • In the "Object Repository Manager Main Window" (described on page 1297), click **Add Insight Object** **Note:** To display the Select Learn Mode dialog box if it does not open when you click these buttons, select **Tools > Options > GUITesting** tab > **Insight** node and then select **Display Select Learn Mode dialog box**.
Relevant tasks	"How to Add an Insight Object to the Object Repository" on page 1214
See also	• "Adding Insight Test Objects" on page 1200. • "Change Test Object Image / Add Insight Test Object Dialog Box" on page 1233

User interface elements are described below:

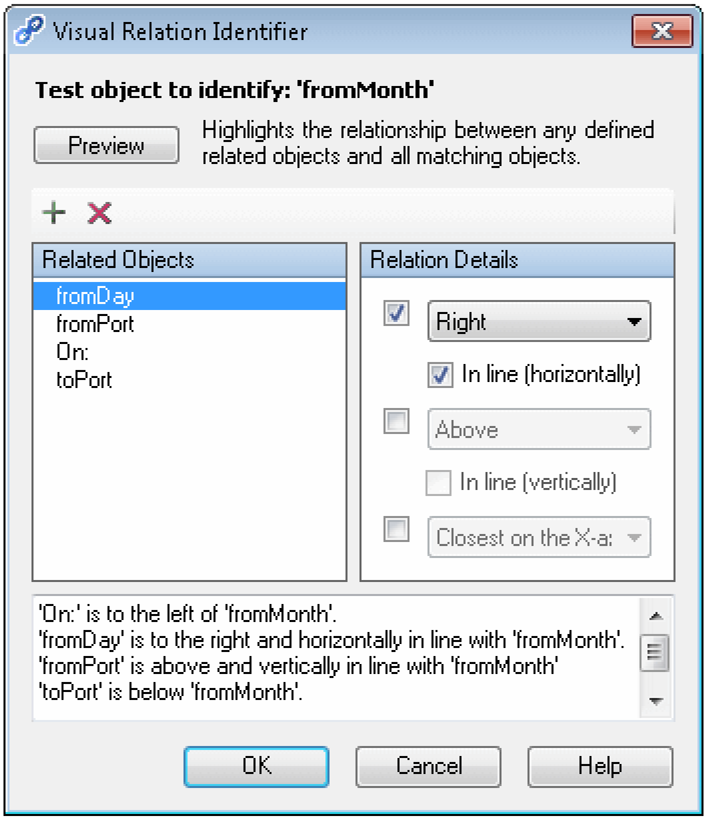
UI Elements	Description
Automatic	Hides UFT and the Object Repository window or Object Repository Manager, and changes the pointer into a pointing hand, enabling you to click on the control that you want to learn in the application.

UI Elements	Description
Manual	Hides UFT and the Object Repository window or Object Repository Manager, and changes the pointer into a crosshair that you can use to draw a rectangle around the control that you want to learn in the application. To help you accurately draw the borders of the control, a magnified view of the mouse location on the screen is displayed in a circle near the crosshair.
Do not show me again	Instructs UFT not to display this dialog box in the future, when adding Insight objects to an object repository, but to use the learning mode that you select this time. To display the Select Learn Mode dialog box again: Select **Tools > Options > GUI Testing** tab > **Insight** node and then select **Display Select Learn Mode dialog box**.

Visual Relation Identifier Dialog Box

Relevant for: GUI tests and components

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



Test object to identify: 'fromMonth'

Highlights the relationship between any defined related objects and all matching objects.

Related Objects:
- fromDay
- fromPort
- On:
- toPort

Relation Details:
- ☑ Right
- ☑ In line (horizontally)
- □ Above
- □ In line (vertically)
- □ Closest on the X-a:

'On:' is to the left of 'fromMonth'.
'fromDay' is to the right and horizontally in line with 'fromMonth'.
'fromPort' is above and vertically in line with 'fromMonth'
'toPort' is below 'fromMonth'.

To access	1. Open the "Object Properties Dialog Box" (described on page 1271) for the test object you want to identify, or select it in the Object Repository window. 2. In the **Visual Relation Identifier Settings** row of the Object Repository window or Object Properties dialog box, click in the **Value** cell. 3. Click the **Browse** button in the text box.
Important information	"Considerations for Working with Visual Relation Identifiers" on page 1256
Relevant tasks	• "How to Maintain Test Objects in Object Repositories" on page 1220 • "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 1225

See also	- "Visual Relation Identifiers" on page 1209 - "How UFT Identifies Objects During a Run Session" on page 1169 - "How UFT Interprets Horizontal and Vertical Visual Relations" on page 1255

User Interface Elements

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

UI Elements	Description
Test object to identify	The name of the test object as it appears in the object repository (read-only).
+	**Add.** Opens the "Select Test Object Dialog Box" (described on page 958), which enables you to add a test object to the **Related Objects** list, either from the object repository or directly from the application. Any object you add from the application is automatically added to the object repository.
✕	**Remove.** Removes the selected related object from the **Related Objects** list. **Note:** Removing the related object from the list does not remove the test object remains from the object repository.

UI Elements	Description
Preview	Highlights the visual relation between all related objects and the objects matching the test object to identify (main UFT window is hidden). While in **Preview** mode, the text to the right of the button displays the number of matching objects. This enables you to test the relation details you are defining without closing the dialog box or running the steps. **Note:** • During the Preview process, the main UFT window is hidden. Click the **Preview** button again to restore the UFT window. • While the **Preview** button is pressed, you can change the relation details and preview the changes without returning to the UFT window. • If UFT is unable to perform the Preview operation, a message box opens. This could be the result of one of the following scenarios: ▪ One or more related objects cannot be found in the object repository. ▪ One or more related objects are already used in a visual relation identifier for another test object. ▪ One or more related objects cannot be uniquely identified in the object repository.
Related Objects	The list of related objects. **Tooltip.** The tooltip for each related object displays the full name. **Note:** If the related object cannot be found in the object repository, an indicating icon is displayed next to the name of that related object, and a tooltip is displayed when you hover the cursor over the icon.
Relation Details	The details of the visual relation for the selected related object. You can select a value from one or more of the relation categories. For details on the available visual relation categories, see "Visual Relation Categories (Relation Details Area)" on the next page.
<Relation description>	Textual description of the currently defined visual relations.

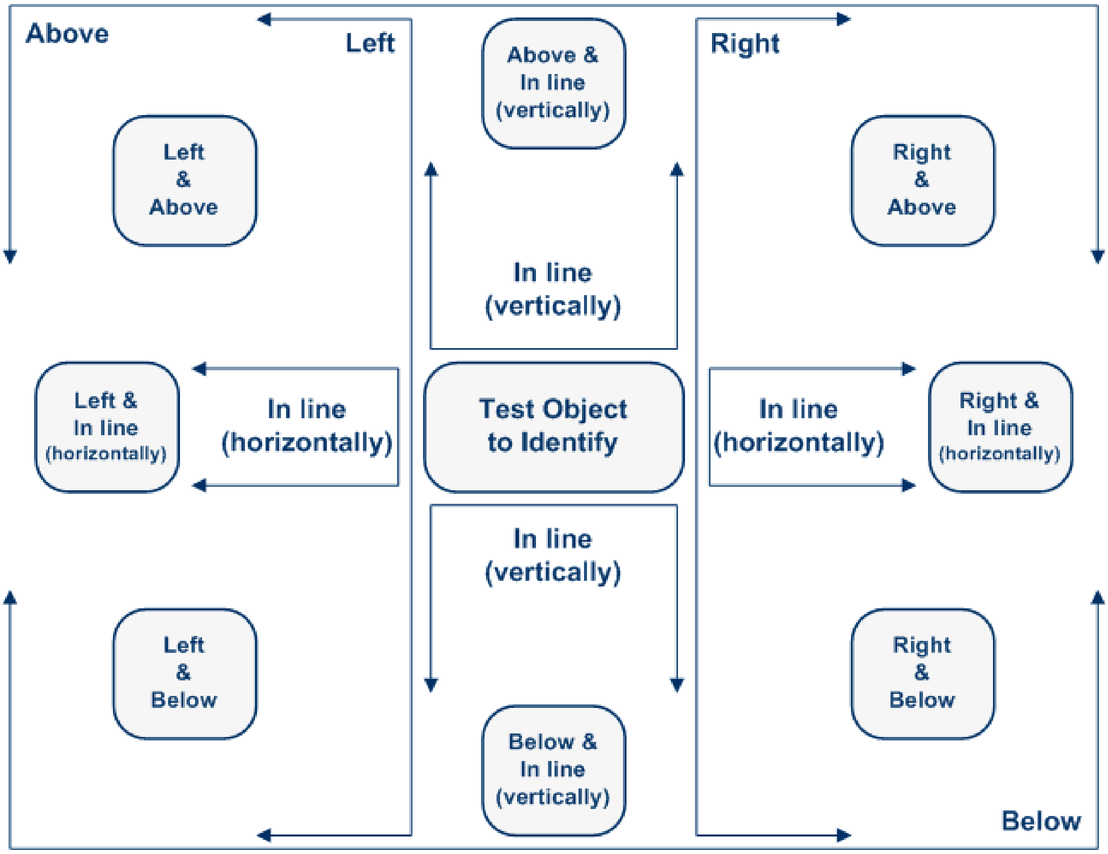
Visual Relation Categories (Relation Details Area)

Category	Description
Horizontal	Enables you to define related objects according to their horizontal location relative to the object to identify. The following options are available: • **Left** • **Right** • **In line (horizontally)** For details on this option, see "How UFT Interprets Horizontal and Vertical Visual Relations" on the next page.
Vertical	Enables you to define related objects according to their vertical location relative to the object to identify. The following options are available: • **Above** • **Below** • **In line (vertically)** For details on this option, see "How UFT Interprets Horizontal and Vertical Visual Relations" on the next page.
Distance and hierarchy	Enables you to define related objects according to their distance or hierarchical location relative to the object to identify. The following options are available: • **Closest on the X-axis** • **Closest on the Y-axis** • **Closest on both axes** • **Contains**

How UFT Interprets Horizontal and Vertical Visual Relations

Relevant for: GUI tests and components

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

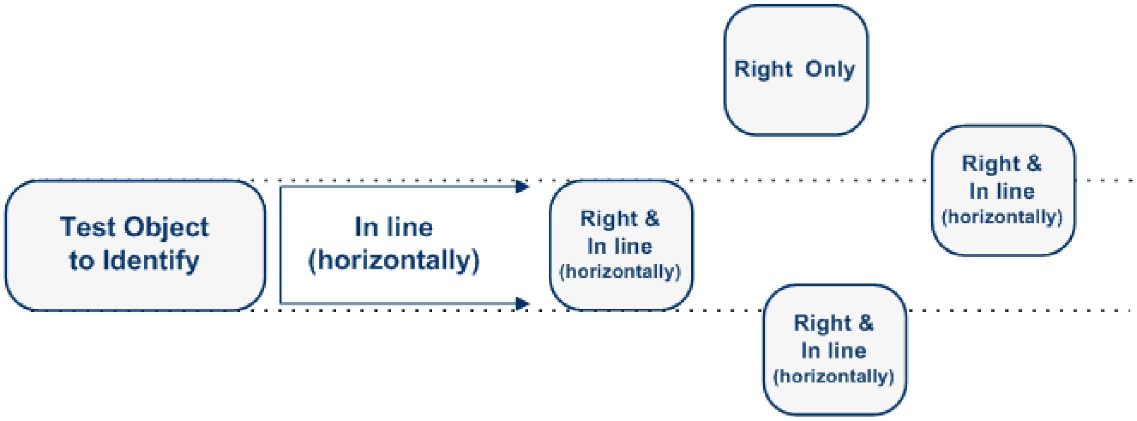


How UFT Identifies In Line Related Objects

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

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

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


Right Only
Right & In line (horizontally)
Test Object to Identify
In line (horizontally)
Right & In line (horizontally)
Right & In line (horizontally)

Considerations for Working with Visual Relation Identifiers

Relevant for: GUI tests and components

Consider the following when working with visual relation identifiers:

- If you define a visual relation identifier for a test object, then that test object's ordinal identifier (if it exists) is not used during the test object identification process (when running steps, highlighting objects in the application, etc.). To indicate this, the **Ordinal identifier** option for that test object is disabled in the Object Repository window.

- If you add a related object that was not previously in the object repository, then the test object for the related object is added to that object repository, even if you click **Cancel** in the Visual Relation Identifier dialog box.

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

- If you delete a test object (A) that is used in the visual relation identifier for another test object (B), you must make sure to remove the deleted test object A from the **Related Object** list for test object B.

- Visual relation identifiers are used only during a run session, or when identifying objects in the application (for example, from the Object Repository window or the Object Spy). Therefore, even if you define related objects for a specific test object, if UFT re-learns the object, it uses only the identification properties that are defined in the "Object Identification Dialog Box" (described on page 1325) for that test object class, as well as an ordinal identifier (if needed). This may cause UFT to learn the same object more than once.

  Example: If you learned the Object1 test object with an ordinal identifier property value of 1, and then you manually remove the ordinal identifier or otherwise significantly modify the description properties (because you are depending on the visual relation identifiers to fine

> tune the description), then the next time a learn session includes this object, UFT will not recognize Object1 as the same object and will learn a new test object with the name Object1_1.

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

  For details on the complete flow that UFT uses to identify objects, see "Object Identification Process Workflow" on page 1188.

- A test object cannot be used as a related object to itself.

- The following test objects cannot act as related objects for another test object's visual relation identifier:

  - A test object that has a visual relation identifier.

  - A child test object for one of its parent objects.

- You can retrieve or replace the visual relation identifier settings of a specific test object during a run session using the **VisualRelationsCollection** object. For details, see the **VisualRelationsCollection** object in the *HP UFT Object Model Reference for GUI Testing*.

Troubleshooting and Limitations - Managing Test Objects

Relevant for: GUI tests and components

For troubleshooting and limitations for learning objects, see "Learning objects, running steps, and recording steps (GUI testing only)" on page 661.

Chapter 44: Working with Object Repositories

Relevant for: GUI tests and components

This chapter includes:

Concepts

Object Repository Window - Overview

Relevant for: GUI tests and components

The Object Repository window displays a tree of all test objects, and all checkpoint and output objects in the current action or component, including all local objects and all objects in associated shared object repositories.

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

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

> **Note:** You modify objects in a shared object repository using the Object Repository Manager. For details, see "Shared Object Repositories" on page 1285.
>
> You can also modify an object in a shared object repository by copying it to the local object repository and then modifying the local copy. For details, see "Local Copies of Objects from Shared Object Repositories " on the next page.

Exporting Local Objects to a Shared Object Repository

Relevant for: GUI tests and components

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

When you export local objects to a shared object repository, the parameters of any parameterized objects are converted to repository parameters, using the same name as the source parameter. The default (mapped) value of each repository parameter is the corresponding source parameter. You can modify the mapping used within your action or component using the "Map Repository Parameters Dialog Box" (described on page 1268). For details on repository parameters, see "Working with Repository Parameters" on page 1287.

> **Tip:** After you export the local objects, you can use the Object Repository Merge Tool to merge the test objects from the shared object repository containing the exported objects with another shared object repository. For details, see "Object Repository Merge Tool" on page 1351.

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

For components: The **Export and Replace Local Objects** option (**File > Export and Replace Local Objects)** is not available.

Local Copies of Objects from Shared Object Repositories

Note: Available from the Object Repository window only.

Relevant for: GUI tests and components

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

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

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

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

Considerations for Copying an Object to the Local Object Repository

- When you copy an object to the local object repository, its parent objects are also copied to the local object repository.

- You cannot copy an object to the local object repository if an object or its parent objects use unmapped repository parameters. You must make sure that all repository parameters are mapped before copying an object to the local object repository.

- If an object or its parent objects are parameterized using one or more repository parameters, the repository parameter values are converted when you copy the object to the local object repository. If the value is a constant value, the property receives the same constant value.

- If you are copying multiple objects to the local object repository, during the copy process you can choose to skip a specific object if it has unmapped repository parameters, or if it has mapped repository parameters whose values you do not want to convert. You can then continue copying the next object from your original selection.

Repository Parameter Value Mappings

Relevant for: GUI tests and components

You use **repository parameters** to specify that certain property values of an object in a shared object repository should be parameterized, but that the actual values should be defined in each action or component associated with the shared object repository. For details on repository parameters, see "Working with Repository Parameters" on page 1287.

Mapping a repository parameter to a value or to a parameter specifies the property values used to identify the test object during a run session. You can specify that the property value is taken from a constant value, or parameterize it using any of the following:

- **For tests:**
  - Data table
  - random number
  - environment
  - test
- **For components:**
  - local parameter
  - component parameter

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

Before you map repository parameters, if you have more than one repository parameter with the same name in different shared object repositories that are associated with the same action or component, the repository parameter from the shared object repository with the highest priority (as defined in the shared object repositories list) is used.

After you map repository parameters, UFT uses the mappings you defined. In addition, changing the priority or default values has no effect after the parameters are mapped.

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

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

Working with Test Objects During a Run Session

Relevant for: GUI tests and components

The first time UFT encounters an object during a run session, it creates a temporary version of the test object for that run session. UFT uses the object description to create this temporary version of the object. For the remainder of the run session, UFT refers to the temporary version of the test object rather than to the test object in the object repository. The Object Repository window is read-only during a run session.

Creating Test Objects During a Run Session

You can use programmatic descriptions to create temporary versions of test objects that represent objects from your application. You can perform operations on those objects without referring to the object repository. For example, suppose an edit box was added to a form on your Web site. You can use a programmatic description to add a statement in a user-defined function or in the Editor (actions only) that enters a value in the new edit box. UFT could then identify the object even though the object was never added to the object repository. For details on programmatic descriptions, see "Programmatic Descriptions" on page 995.

Tasks

How to Export Local Objects to a Shared Object Repository

Relevant for: GUI tests and components

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

This task includes the following steps:

- "Prerequisites " below
- "Select an action (actions only)" below
- "Export the local objects" below
- "Results" below

1. ### Prerequisites

   a. Open the test or component that has the local objects you want to export.

   b. Open the Object Repository window by selecting **Resources > Object Repository** or clicking the **Object Repository** button .

2. ### Select an action (actions only)

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

3. ### Export the local objects

   Select **File > Export Local Objects**, or, for actions only, **File > Export and Replace Local Objects**. The Save Shared Object Repository window opens. For details, see "Save <Resource>/Save <Document> As Dialog Box" on page 164.

4. ### Results

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

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

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

Repository window, the repository location indicates the path and filename of the new shared object repository instead of **Local**.

How to Copy an Object to the Local Object Repository

Relevant for: GUI tests and components

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

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

How to Modify Identification Properties During a Run Session

Relevant for: GUI tests and components

You can modify the properties of the temporary version of the object during the run session without affecting the permanent values in the object repository. To do this, add a **SetTOProperty** statement in a user-defined function, or in your action.

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

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

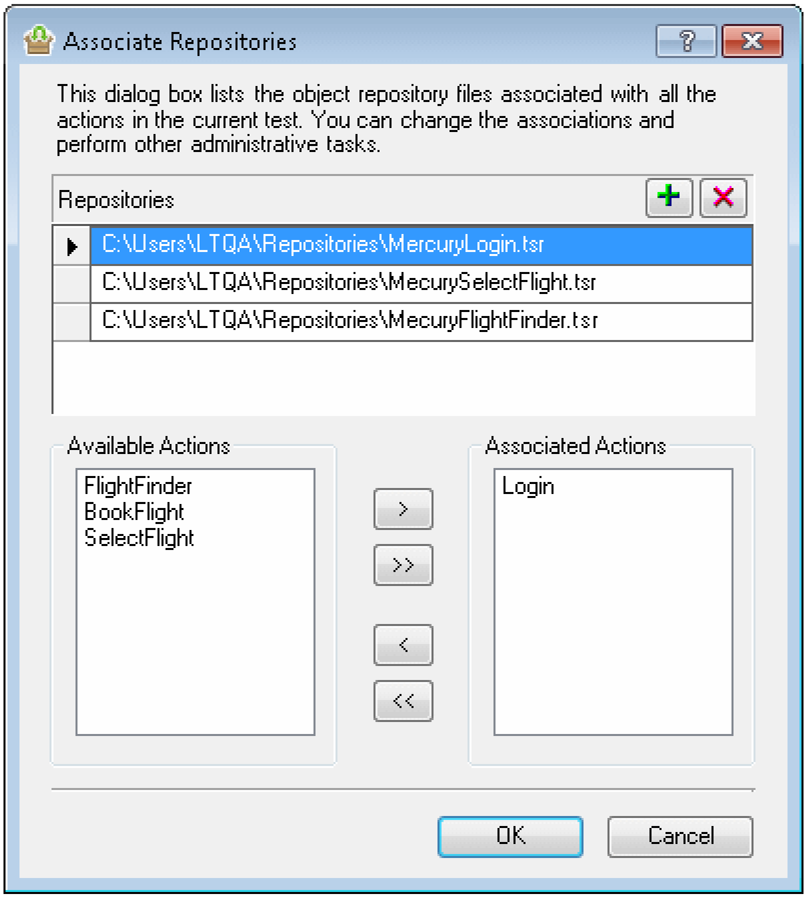
For details, see the *HP UFT Object Model Reference for GUI Testing*.

Reference

Associate Repositories Dialog Box

Relevant for: GUI tests only

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



This dialog box lists the object repository files associated with all the actions in the current test. You can change the associations and perform other administrative tasks.

Repositories
C:\Users\LTQA\Repositories\MercuryLogin.tsr
C:\Users\LTQA\Repositories\MecurySelectFlight.tsr
C:\Users\LTQA\Repositories\MecuryFlightFinder.tsr

Available Actions:
FlightFinder
BookFlight
SelectFlight

Associated Actions:
Login

To access	1. Open the test to which you want to associate your object repositories. 2. Do one of the following: ■ In the UFT main window, select **Resources > Associate Repositories**. ■ In the Object Repository window, select **Tools > Associate Repositories**. ■ In the Object Repository window, click the **Associate Repositories** button .
Important information	• You prioritize the object repositories using the Associated Repositories tab of the Action Properties dialog box (and not the Associate Repositories dialog box). For details, see "Associated Repositories Tab (Action Properties Dialog Box)" on page 906. • You can associate, remove, prioritize, and view the properties of shared object repositories in the Solution Explorer. For details, see "Solution Explorer Pane User Interface" on page 478.

User interface elements are described below:

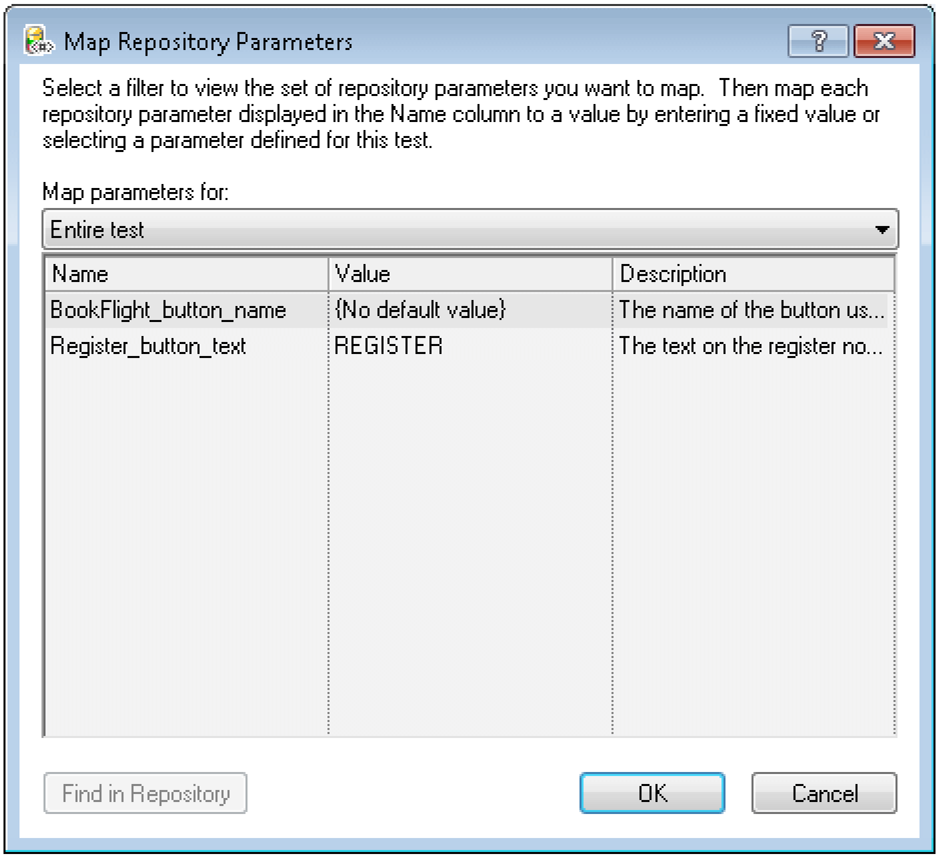
UI Elements	Description
✚	**Add Repository.** Enables you to browse and add shared repositories to be associated with actions. The new object repository is displayed at the bottom of the Repositories list.
✕	**Remove Repository.** Enables you to remove shared repositories and their associations from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

UI Elements	Description
Repositories	The list of available shared repositories. **Tip:** You can modify the name or path of an associated shared object repository without changing the current associations, as follows: • Click a shared object repository and then browse to select a different shared object repository. • Click the shared object repository and modify the name or path directly.
Available Actions	The list of actions to associate to the selected shared repository.
Associated Actions	The list of actions currently associated with the selected shared repository.

Map Repository Parameters Dialog Box

Relevant for: GUI tests and components

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



Select a filter to view the set of repository parameters you want to map. Then map each repository parameter displayed in the Name column to a value by entering a fixed value or selecting a parameter defined for this test.

Map parameters for: Entire test

Name	Value	Description
BookFlight_button_name	{No default value}	The name of the button us...
Register_button_text	REGISTER	The text on the register no...

To access	1. Open the test or component for which you want to map repository parameters. 2. Use one of the following: - Select **Resources > Map Repository Parameters** - In the Errors pane, double-click the **Repository Parameters** row (Relevant only if your test or component contains unmapped repository parameters (repository parameters without a default value)).
See also	"Repository Parameter Value Mappings" on page 1262

User interface elements are described below:

UI Elements	Description
Map parameters for	Enables you to filter the list of parameters that is displayed. You can choose to display: - **All unmapped parameters.** Displays all of the parameters in your test that are not mapped to values. - **Entire test.** (Tests only) Displays all of the parameters in your test (with mapped or unmapped values). - **<Action name> or <Component name>.** (For example, LogIn) Displays all of the parameters in the specified action or component (with mapped or unmapped values).
Name	The name of the repository parameter.
Value	The parameter's current value, if any. This column shows either the new value you defined, or the default value that was defined when the parameter was created. If no default value was defined, then the parameter is currently unmapped, and the text **{No default value}** is shown. You can: - Enter a new constant value or modify an existing constant value by typing directly in the **Value** cell. **Tip:** You can also enter a constant value in the "Value Configuration Options Dialog Box" (described on page 1579) by clicking the parameterization button . - Parameterize the value by clicking in the **Value** cell of the relevant parameter and then clicking the parameterization button . You can parameterize the value for actions, using the Data pane (Global sheet only), random number, environment, or test parameter. You can parameterize the value for components, using a local or component parameter. For details, see "Value Configuration Options Dialog Box" on page 1579. - Reset a parameter to its default value by clicking in the **Value** cell of the relevant parameter and then clicking the **Reset to Default Value** button . The default value, if any, that was defined in the "Add Repository Parameter Dialog Box" (described on page 1308) is displayed in the cell.

UI Elements	Description
Description	A textual description of the parameter, if any.
Find in Repository	Opens the Object Repository window and highlights the first test object in the object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so forth.

Object Properties Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to:

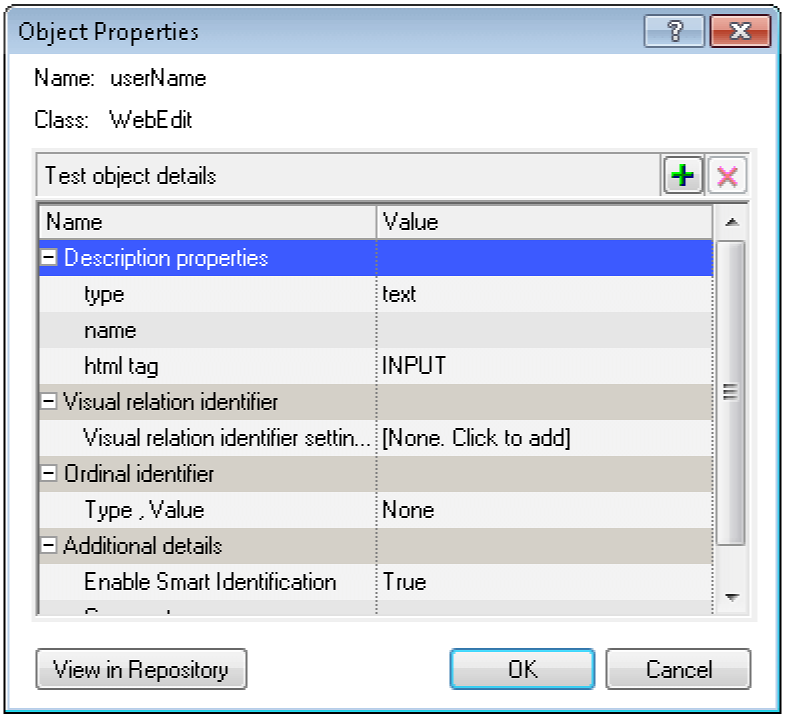
- View identification properties and values for objects in your test or component steps, or in the Active Screen (for actions), as well as other object details.

- Modify the properties and property values used to identify the object (for objects that are stored in the local object repository). You modify the properties and values in the Object Properties Dialog Box in the same way as you modify the test object details in the "Object Repository Window" (described on page 1274). For details, see "Maintaining Identification Properties" on page 1204.

> **Note:**
>
> - For Insight objects, you can view properties such as the ordinal identifier and visual relation identifiers, but not the test object image. To view and modify the test object image, click **View in Repository** to open the Insight test object in the Object Repository window.
>
> - This dialog box is not available for Insight objects in the Active Screen.

There are slight differences in the Object Properties dialog box, depending on whether the selected object is currently stored in the local object repository or in an associated shared object repository.

This section describes options shown in the dialog box for objects in the local object repository. For objects stored in a shared object repository the information is in read-only format.



Name: userName

Class: WebEdit

Name	Value
⊟ Description properties	
type	text
name	
html tag	INPUT
⊟ Visual relation identifier	
Visual relation identifier settin...	[None. Click to add]
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	True

To access	1. Ensure that a GUI action or component is in focus in the document pane. 2. Do use one of the following: **For actions:** ■ In the action, right-click in the step of the object whose properties you want to view and choose **Object Properties.** ■ In the Active Screen, right-click the object whose properties you want to view and choose **View / Add Object**. **For components:** In the component, right-click the step of the object whose properties you want to view and choose **Object Properties**.

User interface elements are described below:

UI Elements	Description
➕	**Add description properties** button.
✖	**Remove selected description properties** button.

UI Elements	Description
Name	The name that UFT assigns to the object. You can change the name of a object in the local object repository. For details, see "Renaming Test Objects " on page 1206.
Class	The class of the object.
Description properties	The properties and property values used to identify the object during a run session. **Tip:** • For details on adding properties to or removing properties from the test object description, see "Add Properties Dialog Box" on page 1231. • For details on specifying a property value as a constant or parameterizing a value, see "Specifying or Modifying Property Values" on page 1205.
Visual relation identifier	A set of definitions that enable you to identify the object in the application according to its neighboring objects in the application. When this option is defined and enabled, the Ordinal identifier option is disabled. For details, see "Visual Relation Identifier Dialog Box" on page 1251. **Note:** If one or more related objects cannot be found in the object repository, indicating text is displayed in the cell.
Ordinal identifier	A numerical value that indicates the object's order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). For details, see "Ordinal Identifier Dialog Box" on page 1247. **Note:** If a visual relation identifier is defined for a specific test object, this option is disabled. For details, see "Considerations for Working with Visual Relation Identifiers" on page 1256.

UI Elements	Description
Additional details	Contains the following options: - **Enable Smart Identification.** Enables you to select **True** or **False** to specify whether UFT should use Smart Identification to identify the test object during the run session if it is not able to identify the object using the test object description. **Note:** This option is available only if Smart Identification properties are defined for the test object's class in the "Object Identification Dialog Box" (described on page 1325). For details, see "Smart Identification" on page 1316. - **Comment.** Enables you to add textual information about the test object.
View in Repository	Opens the Object Repository window and displays the identification properties and values for the selected object.
Add to Repository (actions only)	Displayed only for an object that is not in the object repository. (Available only when you open this dialog box from the Active Screen.) Adds this object to the action's local object repository. After this object is added, the **Add to Repository** button changes to **View in Repository**.

Object Repository Window

Relevant for: GUI tests and components

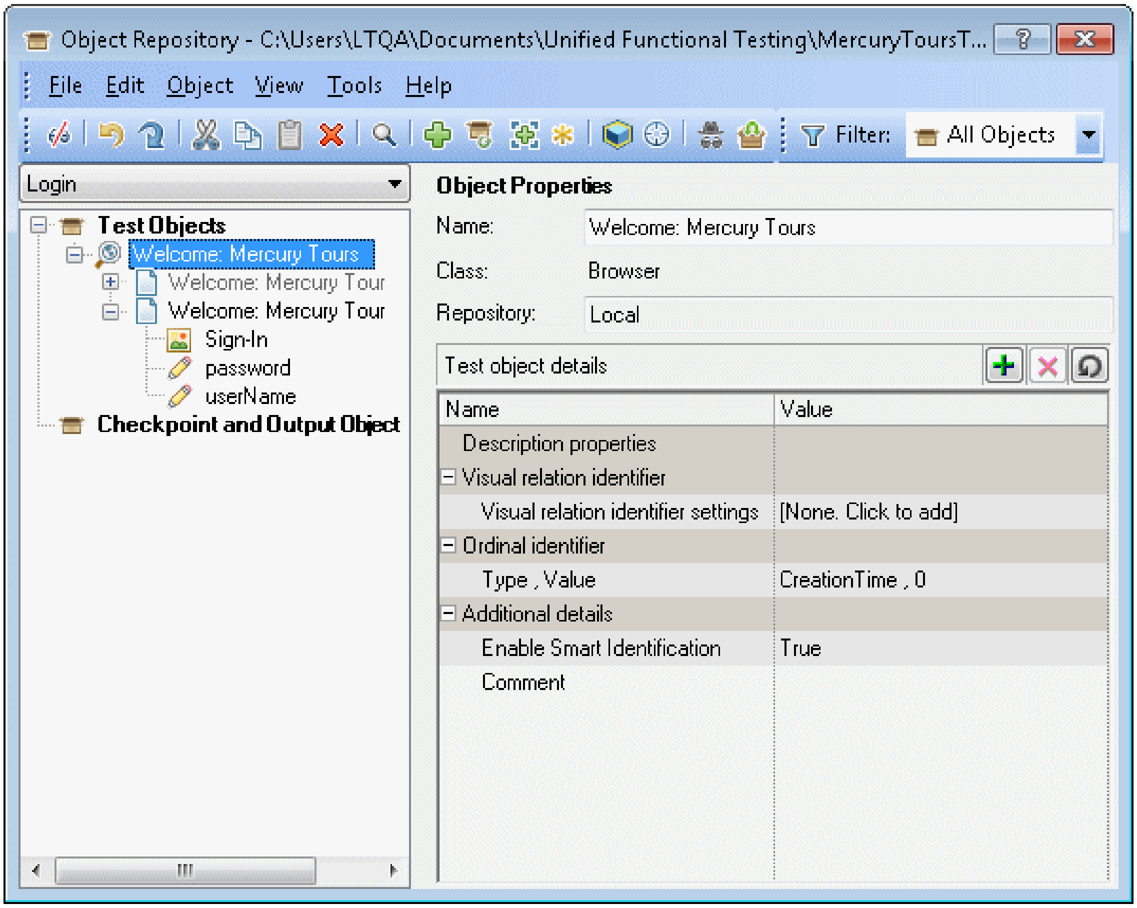
This window enables you to manage identification properties and object repository associations for your action or component.

You can use the Object Repository window to:

- View the object description of any object in the repository (in local and shared object repositories).
- Modify local objects and their properties.
- Add test objects to your local object repository.
- Drag and drop test objects to your test or component. When you drag and drop a test object to your test or component, UFT inserts a step with the default operation for that test object in your test or component.

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

The following image shows the Object Repository window displaying the local object repository of an action, with an Image test object selected. The options on this dialog box differ slightly when displaying a component's object repository, or when selecting other types of objects.



Name: Welcome: Mercury Tours

Class: Browser

Repository: Local

Name	Value
Description properties	
⊟ Visual relation identifier	
Visual relation identifier settings	[None. Click to add]
⊟ Ordinal identifier	
Type , Value	CreationTime , 0
⊟ Additional details	
Enable Smart Identification	True
Comment	

To access	1. 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. 2. Do one of the following: ▪ **UFT main window:** Click the **Object Repository** button , or choose **Resources > Object Repository** ▪ **Solution Explorer pane:** Double-click an object repository, or right-click an object repository and choose **Open Repository** ▪ **GUI Test:** Right-click an action in the canvas and choose **Object Repository** ▪ **Toolbox pane:** Right-click a test object and choose **Open Resource** ▪ **Record toolbar:** Click the **Object Repository** button during a recording session.
Important information	• Local objects are editable (black). Objects from a shared object repository are read-only format (gray). • You can modify checkpoint and output value details for objects saved in the local object repository. • You cannot drag and drop checkpoint and output objects from the Object Repository window to your testing document. • You can copy an object from a shared object repository to the local object repository, and then modify it. • Test objects for environments that are not installed/loaded with UFT are displayed with a question mark icon. • The Object Repository window is read-only during a record or run session.

Relevant tasks	**Primary tasks:** - "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217 - "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 1225 **Related tasks:** - "How to Export Local Objects to a Shared Object Repository" on page 1264 - "How to Copy an Object to the Local Object Repository" on page 1265
See also	- For an overview of this window, see "Object Repository Window - Overview" on page 1260. - For details on **dragging and dropping test objects from other locations**, see "Toolbox Pane Overview" on page 506 and "How to Manage Objects in Shared Object Repositories" on page 1294. - For details on **modifying the properties of a test object during a run session**, see "Working with Test Objects During a Run Session" on page 1263. - For details on **viewing and modify object properties from other locations**, see "Maintaining Identification Properties" on page 1204.

User interface elements are described in the following sections:

- "Object Repository Window - Edit Toolbar" below
- "Object Repository Window - Filter Toolbar" on page 1279
- "Object Repository Window Options" on page 1280
- "Object Repository Window - Object Details Area" on page 1281
- "Object Repository Window - Test Object Image Area" on page 1282

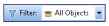
Object Repository Window - Edit Toolbar

Button	Name	Description
	Compact View	**Compact View** mode displays only the object repository tree, while **Full View** mode displays the object repository tree together with the object details area.
	Full View	

Button	Name	Description
↶	**Undo**	All changes you make to a local object are automatically updated in all steps that use the local object as soon as you make the change. You can use the **Edit > Undo** and **Edit > Redo** menu options or **Undo** and **Redo** toolbar buttons to cancel or repeat your changes. After you save the current test or component, you cannot undo or redo operations that were performed before the save operation.
↷	**Redo**	
✂	**Cut**	Cuts the selected object from the object repository tree. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
	Paste	Pastes the object in the clipboard into the object repository tree as a child of the object selected in the tree. Bottom level objects cannot contain children. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
📋	**Copy**	Copies the selected object from the object repository tree into the clipboard. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
✖	**Delete**	Deletes the selected object from the object repository tree.
🔍	**Find & Replace**	Finds and replaces an object in the object repository. For details, see "Find and Replace Dialog Box (Object Repository)" on page 1245.
➕	**Add Objects to Local**	Enables you to add objects to the local object repository. For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199.
	Update from Application	Enables you to update the identification properties from an object in the application. For details, see "Updating Identification Properties from an Object in Your Application" on page 1206.
	Add Insight Object to Local	Enables you to add an Insight object to the local object repository. For details, see "Adding Insight Test Objects" on page 1200.
✳	**Define New Test Objects**	Enables you to define a new test object. For details, see "Define New Test Object Dialog Box" on page 1240.
	Highlight in Application	Highlights the selected object in the object repository tree, in the application. For details, see "Highlighting an Object in Your Application" on page 1203.
⌖	**Locate in Repository**	Enables you to select an object in the application you are testing and highlight the test object in the object repository. For details, see "Locating a Test Object in the Object Repository" on page 1204.

Button	Name	Description
	Object Spy	Enables you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and operations of the test object that uses to represent that object. You can also add an object to the local object repository and highlight an object in the application. For details, see "Object Spy Dialog Box " on page 1189.
	Associate Repositories (actions only)	Enables you to manage the shared object repository associations of your action. For details, see "Associate Repositories Dialog Box" on page 1266.

Object Repository Window - Filter Toolbar

UI Element	Description
	You can use the Filter toolbar to filter the objects shown in the Object Repository window. You can choose to show objects that meet one of the following criteria: • All objects in the current component or in the selected action, including all local objects and all objects in any shared object repositories associated with the selected action. • Only the local objects in the current component or in the selected action. • Only the objects in a specific shared object repository associated with the current action or component. To filter the Object Repository window: In the **Filter** toolbar list, select one of the following options: • **All Objects** • **Local Objects** • The name of a specific shared object repository associated with the current action or component The object repository tree is filtered to display only the objects from the location that you selected. The title bar of the Object Repository window indicates the current filter.

Object Repository Window Options

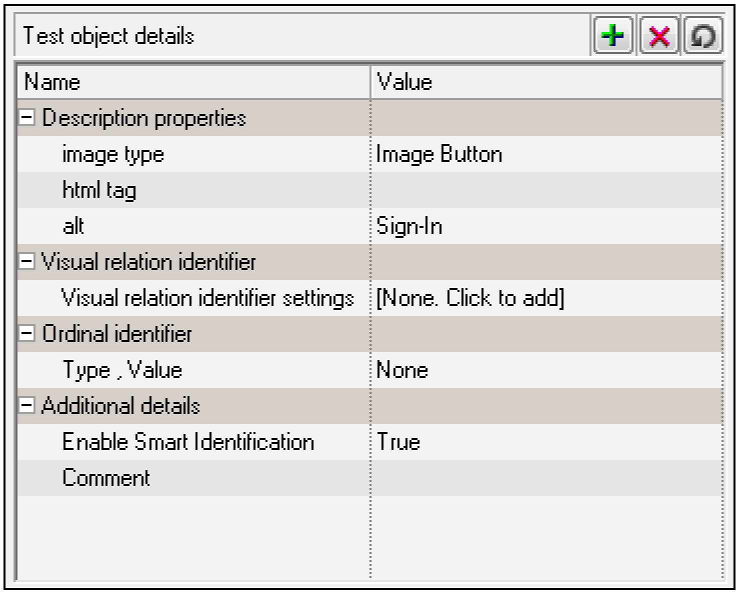
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Action (for actions)	Enables you to select the action whose objects you want to view.
Business Component (for components)	Indicates that the current testing document is a business component.
<Object repository tree>	Displays all of the test objects, checkpoint objects, and output objects in the local and shared object repositories associated with in the current component or in the selected action. You can filter the objects shown in the object repository tree. For details, see "Object Repository Window - Filter Toolbar" on the previous page. **Note:** If there are test objects in different associated object repositories with the same name, object class, and parent hierarchy, the object repository tree shows only the first one it finds based on the priority order defined. For details, see: • **For actions:** "Associated Repositories Tab (Action Properties Dialog Box)" on page 906 • **For components:**"Object Repositories Pane (Application Area)" on page 2105.
Name	The name that UFT assigns to the object. You can change the name of a object in the local object repository. For details, see "Renaming Test Objects " on page 1206.
Class	The class of the object.
Repository	The location (file name and path) of the object repository in which the object is located. If the object is located in the local object repository, **Local** is displayed.
<Object details area>	Displays and enables you to modify one of the following: • The properties and property values used to identify a test object during a run session • The properties of a checkpoint or output object. For details, see the "Object Repository Window - Object Details Area" (described on page 1281).

UI Elements	Description
Test object image area	Displays the image used to identify the object and enables you to modify it. For details, see the "Object Repository Window - Test Object Image Area" (described on page 1282).

Object Repository Window - Object Details Area

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



Name	Value
⊟ Description properties	
image type	Image Button
html tag	
alt	Sign-In
⊟ Visual relation identifier	
Visual relation identifier settings	[None. Click to add]
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	True
Comment	

For test objects:

UI Elements	Description
Description properties	The properties and property values used to identify the object during a run session. **Tip:** • For details on adding properties to or removing properties from the test object description, see "Add Properties Dialog Box" on page 1231. • For details on specifying a property value as a constant or parameterizing a value, see "Specifying or Modifying Property Values" on page 1205.

UI Elements	Description
Visual relation identifier	A set of definitions that enable you to identify the object in the application according to its neighboring objects in the application. When this option is defined and enabled, the Ordinal identifier option is disabled. For details, see "Visual Relation Identifier Dialog Box" on page 1251. **Note:** If one or more related objects cannot be found in the object repository, indicating text is displayed in the cell.
Ordinal identifier	A numerical value that indicates the object's order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). For details, see "Ordinal Identifier Dialog Box" on page 1247. **Note:** If a visual relation identifier is defined for a specific test object, this option is disabled. For details, see "Considerations for Working with Visual Relation Identifiers" on page 1256.
Additional details	Contains the following options: • **Enable Smart Identification.** Enables you to select **True** or **False** to specify whether UFT should use Smart Identification to identify the test object during the run session if it is not able to identify the object using the test object description. **Note:** This option is available only if Smart Identification properties are defined for the test object's class in the "Object Identification Dialog Box" (described on page 1325). For details on Smart Identification, see "Smart Identification" on page 1316. • **Comment.** Enables you to add textual information about the test object.

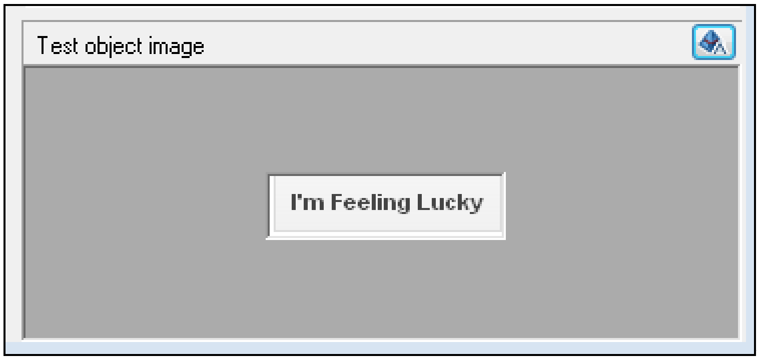
For checkpoints: The object details area contains the same information as the Checkpoint Properties dialog box. For details, see "Checkpoint Properties Dialog Box" on page 1432.

For output objects: The object details area contains the same information as the Output Value Properties dialog box. For details, see "Output Value Properties Dialog Box" on page 1498.

Object Repository Window - Test Object Image Area

The **Test Object Image** area below the test object details area is available only for Insight test objects.

It displays the image that UFT uses to identify the object in the application. To modify the image, click the **Change Test Object Image** button in the title bar of this area. The "Change Test Object Image / Add Insight Test Object Dialog Box"(described on page 1233) opens. In this dialog you can change the test object image and also the default location to click in the object when performing methods on the object.



s

Troubleshooting and Limitations - Object Repositories

Relevant for: GUI tests and components

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

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

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

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

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

Chapter 45: Shared Object Repositories

Relevant for: GUI tests and components

This chapter includes:

Concepts

Shared Object Repositories Overview

Relevant for: GUI tests and components

A shared object repository contains information that enables UFT to identify the objects in your application. UFT enables you to maintain the reusability of your tests or components by storing all the information regarding your test objects in shared object repositories.

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

When objects in your application change, the Object Repository Manager provides a single, central location in which you can update test object information for multiple tests or components.

Advantages of Shared Object Repositories

- You can use the same shared object repository with multiple actions or components, instead of saving objects directly with an action or a component in a local object repository. This enables them to be accessed from multiple actions or components.

- You can use multiple shared object repositories with each action or component.

- If your shared object repositories are stored in ALM, you can apply version control to them. For details, see "Version Control in ALM" on page 794.

For details on associating shared object repositories, see:

- **For tests:**"Add an object repository file to the current solution (GUI testing only)" on page 476

- **For components:** "Object Repositories Pane (Application Area)" on page 2105

> **Note:** Instead of, or in addition to, shared object repositories, you can choose to store all or some of the objects in a local object repository for each action or component, using the Object Repository window. For details on local object repositories, see "Managing Test Objects in Object Repositories" on page 1198.

This section also includes:

Importing and Exporting Shared Object Repositories Using XML

Relevant for: GUI tests and components

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

You can view the required format for the shared object repository in the *HP Unified Functional Testing Object Repository Schema Help* (**Help > HP UFT GUI Testing Advanced References > Object Repository Schema Reference**), or by exporting a saved shared object repository.

You can import and export files from and to the file system or an ALM project (if UFT is connected to ALM).

For details, see:

- "Import a shared object repository from XML" on page 1294
- "Export a shared object repository to XML" on page 1294

Working with Repository Parameters

Relevant for: GUI tests and components

Repository parameters enable you to specify that certain property values should be parameterized, but leave the actual parameterization to be defined in each test or component that is associated with the shared object repository that contains the parameterized identification property values.

Repository parameters are useful when you want to create and run tests and components on an object that changes dynamically. An object may change dynamically if it is frequently updated in the application, or if its property values are set using dynamic content, for example, from a database.

Example

> Suppose you have a button whose text property value changes in a localized application depending on the language of the user interface. You can parameterize the **name** property value using a repository parameter, and then in each test or component that uses the shared object repository you can specify the location from which the property value should be taken. For example, in one test or component that uses this shared object repository you can specify that the property value comes from an environment variable or a component parameter, respectively. In another test or component it can come from the Data pane or a local parameter, respectively. In a third test or component you can specify it as a constant value.

You define all the repository parameters for a specific shared object repository using the "Manage Repository Parameters Dialog Box" (described on page 1306).

Considerations for Working with Repository Parameters

- When you delete a repository parameter that is used in a test object definition, the identification property value remains mapped to the parameter, even though the parameter no longer exists. Therefore, before deleting a repository parameter, you should make sure that it is not used in any test object descriptions, otherwise tests or components that have steps using these test objects will fail when you run them.

- When you open a test or component that uses a shared object repository with a repository parameter that has no default value, an indication that there is a repository parameter that needs mapping is displayed in the Errors pane. You can then map the repository parameter as needed in the test or component. You can also map repository parameters that have default values, and change mappings for repository parameters that are already mapped. For details on mapping repository parameters, see "Unmapped Shared Object Repository Parameter Values" on page 368.

Managing Shared Object Repositories Using Automation

Relevant for: GUI tests and components

UFT provides an Object Repository automation object model that enables you to manage UFT shared object repositories and their contents from outside of UFT. The automation object model enables you to use a scripting tool to access UFT shared object repositories via automation.

Just as you use the UFT automation object model to automate your UFT operations, you can use the objects and methods of the Object Repository automation object model to write scripts that manage shared object repositories, instead of performing these operations manually using the Object Repository Manager. For example, you can add, remove, and rename test objects; import from and export to XML; retrieve and copy test objects; and so forth.

After you retrieve a test object, you can manipulate it using the methods and properties available for that test object class. For example, you can use the GetTOProperty and SetTOProperty methods to retrieve and modify its properties. For details on available test object methods and properties, see the *HP UFT Object Model Reference for GUI Testing*.

Automation programs are especially useful for performing the same tasks multiple times or on multiple shared object repositories. You can write your automation scripts in any language and development environment that supports automation. For example, you can use VBScript, JavaScript, Visual Basic, Visual C++, or Visual Studio .NET. For general information on controlling UFT using automation, see "UFT Automation Scripts" on page 1155.

Using the Unified Functional Testing Object Repository Automation Reference

The Unified Functional Testing Object Repository Automation Reference is a Help file that provides detailed descriptions, syntax information, and examples for the objects and methods in the UFT shared object repository automation object model.

The Help topic for each automation object includes a list and description of the methods associated with that object. Method Help topics include detailed description, syntax, return value type, and argument value information.

You can open the *Unified Functional Testing Object Repository Automation Reference* from the main **UFT Help** menu (**Help > HP UFT GUI Testing Advanced References > Object Repository Automation Reference**).

> **Note:** The syntax and examples in the Help file are written in VBScript-style. If you are writing your automation program in another language, the syntax for some methods may differ slightly from what you find in the corresponding Help topic. For details on syntax for the language you are using, see the documentation included with your development environment or to general documentation for the programming language.

Considerations for Working with Shared Object Repositories

Relevant for: GUI tests and components

Consider the following when working with shared object repositories:

Opening, Modifying, and Saving Shared Object Repositories

- **Enabling editing.** If you opened the shared object repository in read-only mode, you must enable editing for the shared object repository before you can modify it. This locks the shared object repository and prevents it from being modified simultaneously by multiple users. For details on enabling editing, see "How to Create and Manage Shared Object Repositories" on page 1291.

- **Unlocking.** When you enable editing for a shared object repository, the shared object repository is locked so that it cannot be modified by other users. To enable other users to modify the shared object repository, you must first unlock it (by disabling edit mode, or by closing it). If a shared object repository is already locked by another user, if it is saved in read-only format, or if you do not have the permissions required to open it, you cannot enable editing for it.

- **Applying changes.** All changes you make to a shared object repository are automatically updated in all tests or components open on the same computer that use the shared object repository as soon as you make the change—even if you have not yet saved the shared object repository with your changes.

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

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

- **Merging.** You can modify a shared object repository by merging it with another shared object repository. When you merge two shared object repositories, a new shared object repository is created, containing the content of both shared object repositories. If you merge a shared object repository with a local object repository, the shared object repository is updated with the content of the local object repository. For details, see "Object Repository Merge Tool" on page 1351.

Managing Objects in Shared Object Repositories

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

- The following table describes UFT behavior in cases of duplicate objects in shared object repositories:

If...	UFT Uses...
An object with the same name and description is located in both the local object repository and in a shared object repository that is associated with the same action or component.	The local object definition.
An object with the same name and description is located in more than one shared object repository, and these shared object repositories are all associated with the same action or component.	The object definition from the first occurrence of the object, according to the order in which the shared object repositories are associated with the action or component.

General Tips and Guidelines

- If your test or component contains test objects from environments that are not installed/loaded with UFT, the test objects are displayed with a question mark in the shared object repository.

- When you modify a shared object repository, an asterisk (*****) is displayed in the title bar until the object repository is saved.

- You can use the **Edit > Undo** and **Edit > Redo** options to cancel or repeat your changes as necessary. The **Undo** and **Redo** options are related to the active document. When you save a shared object repository, you cannot undo and redo operations that were performed on that file before the save operation.

- You perform search operations in the shared object repository the same way you perform them in the local object repository. For details, see "Locating Objects" on page 1202.

Tasks

How to Create and Manage Shared Object Repositories

Relevant for: GUI tests and components

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

This task includes the following steps:

- "Prerequisites and considerations" below
- "Create a new shared object repository" below
- "Open a shared object repository" on the next page
- "Enable editing for a shared object repository" on the next page
- "Save a shared object repository" on the next page
- "Close shared object repositories" on page 1293
- "Associate a shared object repository with actions or components" on page 1293
- "Merge object repositories into shared ones" on page 1293
- "Import a shared object repository from XML" on page 1294
- "Export a shared object repository to XML" on page 1294

Prerequisites and considerations

- If you want to edit a shared object repository stored in the file system, and the shared object repository was last modified using a version of QuickTest earlier than version 9.0, UFT must convert it to the current format before you can edit it. If you do not want to convert it, you can view it in read-only format. After the file is converted and saved, you cannot use it with earlier versions of QuickTest.
- If you are working with shared object repositories that are stored in an ALM project, you must connect to ALM either from UFT or from the Object Repository Manager by choosing **ALM > ALM Connection** or clicking the **ALM Connection** button . For details, see "ALM Connection Dialog Box" on page 737.

Create a new shared object repository

Open the Object Repository Manager and create a new shared object repository, as described in "Object Repository Manager Main Window" on page 1297.

Open a shared object repository

Do one of the following:

- In the "Object Repository Manager Main Window" (described on page 1297), select **File > Open** to open the Open Shared Object Repository dialog (for object repositories stored on the file system or an ALM project.

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

  For a user interface description, see "Open/New <Document>/<Resource> Dialog Box" on page 156.

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

- From the Solution Explorer, double-click the name of a shared object repository or right-click and select **Open Repository**.

Enable editing for a shared object repository

Select **File > Enable Editing** or click the **Enable Editing** button ✎ . The shared object repository becomes editable. For details, see "Object Repository Manager Toolbar Buttons" on page 1300.

For considerations on enabling editing, see "Considerations for Working with Shared Object Repositories" on page 1289.

Save a shared object repository

Save the shared object repository, as described in "Object Repository Manager Toolbar Buttons" on page 1300. If the file has already been saved, the changes you made are saved. If the file has not yet been saved, the Save Shared Object Repository window opens.

For details, see "Save <Resource>/Save <Document> As Dialog Box" on page 164.

For considerations on saving shared object repositories, see "Considerations for Working with Shared Object Repositories" on page 1289.

> **Note about relative and absolute paths:** When you specify a path to a resource in the file system, UFT checks if the path, or a part of the path, exists in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node). If the path exists, you are prompted to define the path using only the relative part of the path you entered. If the path does not exist, you are prompted to add the resource's location path to the Folders pane and define the path relatively.
>
> For details, see:
>
> - **For actions:** "Relative Paths in UFT for GUI Testing" on page 130
>
> - **For components:** "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544.

If you are working with the Resources and Dependencies model with Quality Center 10.00 or ALM, you should specify an absolute Quality Center or ALM path. For details, see "Relative Paths and ALM" on page 758.

Close shared object repositories

Do one of the following:

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

- To close all open shared object repositories, select **File > Close All Windows**. All open shared object repositories close and are automatically unlocked. If you made changes that are not yet saved, you are prompted to do so before the files close.

  Note: If you close UFT, the Object Repository Manager also closes. If you have made changes that are not yet saved, you are prompted to do so before the Object Repository Manager closes.

Associate a shared object repository with actions or components

You can associate the shared object repository with one or more actions or components (via the component's application area) from within UFT.

If you want to associate the shared object repository with an application area so that it can be accessed by components, you must save it to your ALM project.

For details on associating shared object repositories, see:

- **For actions:**"Add an object repository file to the current solution (GUI testing only)" on page 476

- **For application areas:**"Object Repositories Pane (Application Area)" on page 2105.

Merge object repositories into shared ones

Using the Object Repository Merge tool, you can merge objects from the local object repository of one or more actions or components to a shared object repository using the **Update from Local Repository** option in the Object Repository Manager (**Tools > Update from Local Repository**).

For example, you may have learned objects locally in a specific action in your test or in a specific component and want to move them to the shared object repository so they are available to all actions in different tests that use that shared object repository, and all components that use it.

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

For details, see "Object Repository Merge Tool" on page 1351.

Import a shared object repository from XML

You can import an XML file (created using the required format) as a shared object repository. The XML file can either be a shared object repository that you exported to XML format using the Object Repository Manager, or an XML file created using a tool such as UFT Siebel Test Express or a custom built utility. You must adhere to the XML structure and format.

To import from XML:

1. In the Object Repository Manager, select **File > Import from XML**. The "Open/New <Document>/<Resource> Dialog Box" (described on page 156) opens.

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

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

Export a shared object repository to XML

You can export the objects in a shared object repository to an XML file. This enables you to edit it using any XML editor, and also enables you to save it in an accessible, versatile format.

To export to XML:

1. Make sure that the shared object repository whose objects you want to export is the active window.

2. Make sure that the shared object repository is saved.

3. In the Object Repository Manager, select **File > Export to XML**. The "Save <Resource>/Save <Document> As Dialog Box" (described on page 164) opens.

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

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

How to Manage Objects in Shared Object Repositories

Relevant for: GUI tests and components

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

> **Note:** Many of the shared object repository operations you can perform in the Object Repository Manager are done in a similar way to how you modify objects stored in a local

object repository (using the Object Repository window).

For this reason, many of the procedures are described in "Managing Test Objects in Object Repositories" on page 1198, and are only referenced here. This task contains operations that you can perform only in the Object Repository Manager.

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

This task includes the following steps:

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

Prerequisites

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

Manage objects in a shared object repository

You can perform the following operations to manage objects in a shared object repository, as described in "Managing Test Objects in Object Repositories" on page 1198:

- "How to Add a Test Object to an Object Repository" on page 1211
- "How to Add an Insight Object to the Object Repository" on page 1214
- "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217
- "How to Locate an Object in an Object Repository" on page 1219
- "How to Maintain Test Objects in Object Repositories" on page 1220
- "How to Copy an Object to the Local Object Repository" on page 1265

Add test objects using the Navigate and Learn option

1. Select **Object > Navigate and Learn** or press F6. The **Navigate and Learn** toolbar opens. For a user interface description, see "Navigate and Learn Toolbar" on page 1302.

   Note: You cannot learn Insight test objects using this option.

2. Click the parent object (for example, Browser, Dialog, Window) you want to add to the shared object repository to focus it. The **Learn** button in the toolbar is enabled.

3. Click the **Learn** button or focus the **Navigate and Learn** toolbar and press ENTER. A flashing highlight surrounds the focused window and the object and its descendants are added to the shared object repository according to the defined filter.

4. When you finish adding the required objects to the shared object repository, click the **Close** button in the **Navigate and Learn** toolbar or press ESC. The **Navigate and Learn** toolbar closes and the Object Repository Manager is redisplayed, showing the objects you just added to the shared object repository.

Manage repository parameters

Use the "Manage Repository Parameters Dialog Box" (described on page 1306).

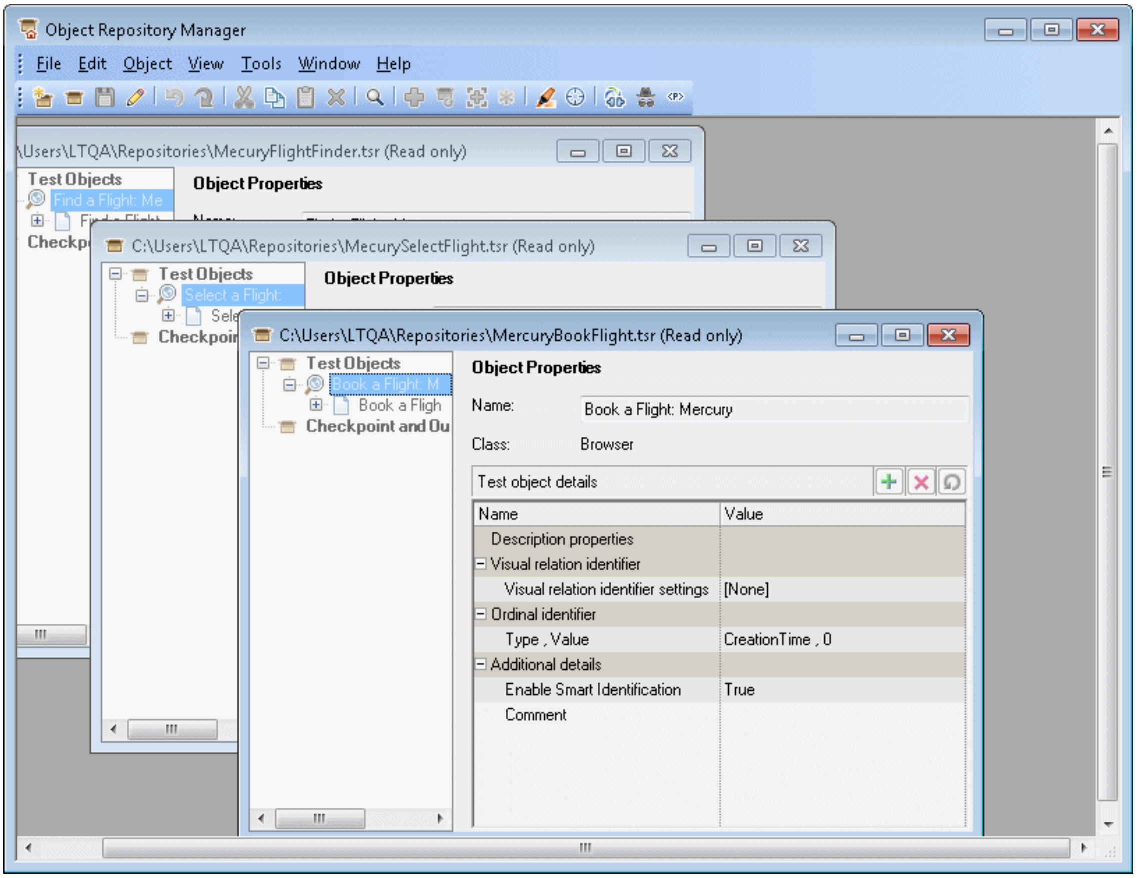
Reference

Object Repository Manager Main Window

Relevant for: GUI tests and components

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

The options available when specifying property values for objects in shared object repositories are different from those available when specifying properties for objects in local repositories.


Object Repository Manager
File Edit Object View Tools Window Help
\Users\LTQA\Repositories\MecuryFlightFinder.tsr (Read only)
Test Objects
Find a Flight: Me
Checkp
Object Properties
C:\Users\LTQA\Repositories\MecurySelectFlight.tsr (Read only)
Test Objects
Select a Flight:
Sele
Checkpoi
Object Properties
C:\Users\LTQA\Repositories\MercuryBookFlight.tsr (Read only)
Test Objects
Book a Flight: M
Book a Fligh
Checkpoint and Ou
Object Properties
Test object details

Name: Book a Flight: Mercury

Class: Browser

Name	Value
Description properties	
Visual relation identifier	
Visual relation identifier settings	[None]
Ordinal identifier	
Type , Value	CreationTime , 0
Additional details	
Enable Smart Identification	True
Comment	

To access	Do one of the following: • Select **Resources > Object Repository Manager**. • In the Solution Explorer, double-click a shared object repository.
Important information	See "Considerations for Working with the Object Repository Manager" on the next page

Relevant tasks	• "How to Create and Manage Shared Object Repositories" on page 1291 • "How to Manage Objects in Shared Object Repositories" on page 1294
See also	• "Shared Object Repositories Overview" on page 1286 • "Considerations for Working with Shared Object Repositories" on page 1289 • "New Property Dialog Box" on page 1232 • "Ordinal Identifier Dialog Box" on page 1247

Considerations for Working with the Object Repository Manager

- You can open as many shared object repositories as you want.

- Each shared object repository opens in a separate document window. You can then resize, maximize, or minimize the windows to arrange them as you require to copy, drag, and move objects between different shared object repositories, as well as perform operations on a single object repository.

- While the Object Repository Manager is open, you can continue working with other UFT windows.

- You cannot add checkpoint or output value objects to a shared object repository via the Object Repository Manager. They are added to the local object repository as the test object they are performed on. You can then upload them to the shared object repository, if needed. For details, see "How to Update a Shared Object Repository From a Local Object Repository" on page 1358.

- When you choose a menu item or click a toolbar button in the Object Repository Manager, the operation you select is performed on the shared object repository whose window is currently active (in focus).

- If UFT is connected to an ALM project with version control enabled, you can view and manage versions of your shared object repositories, view comparisons of two shared object repository versions, and view baseline history. For details, see "Version Control in ALM" on page 794 and "Viewing and Comparing Versions of UFT Assets" on page 770.

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

User interface elements are described in the sections below:

- "Object Repository Document Window" on the next page

- "Object Details Area (Test Objects)" on the next page

- "Test Object Image Area (Insight Test Objects)" below
- "Object Details Area (Checkpoint Objects)" on the next page
- "Object Details Area (Output Value Objects) " on the next page
- "Object Repository Manager Toolbar Buttons" on the next page

Object Repository Document Window

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

UI Element	Description
Title bar	Displays the name and file path of the shared object repository currently active (in focus).
Object Repository tree	Located on the left side of the Object Repository window, and contains all objects in the shared object repository. Test objects of environments that are not installed with UFT are displayed with a question mark icon in the test object tree.
Name	Specifies the name that UFT assigns to the selected object. You can change the object name. For details, see "Renaming Test Objects " on page 1206.
Class	Specifies the class of the selected object.

Object Details Area (Test Objects)

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

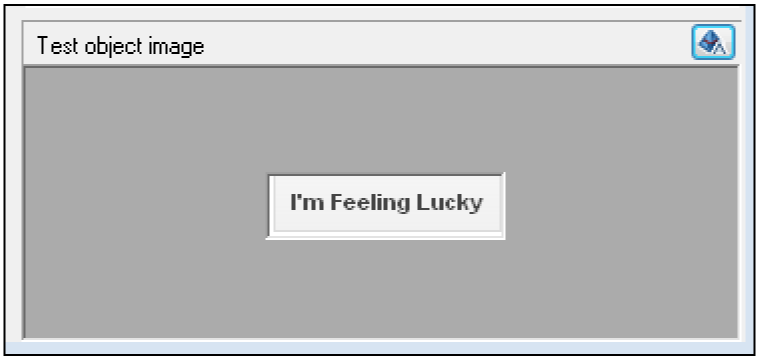
For details on key functionality of this area, see:

- "Updating Identification Properties from an Object in Your Application" on page 1206
- "Restoring Default Mandatory Properties for a Test Object" on page 1206
- "How to Maintain Test Objects in Object Repositories" on page 1220
- "Renaming Test Objects " on page 1206
- "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 1225

Test Object Image Area (Insight Test Objects)

The **Test Object Image** area below the test object details area is available only for Insight test objects.

It displays the image that UFT uses to identify the object in the application. To modify the image, click the **Change Test Object Image** button in the title bar of this area. The Change Test Object Image dialog box (described on page 1233) opens. In this dialog you can change the test object image and also the default location to click in the object when performing methods on the object.



Object Details Area (Checkpoint Objects)

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

For details on specifying and modifying values for properties of a checkpoint object, see the "Checkpoint Properties Dialog Box" on page 1432

Object Details Area (Output Value Objects)

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

For details on specifying and modifying values for properties of an output object, see the "Output Value Properties Dialog Box" on page 1498.

Object Repository Manager Toolbar Buttons

Relevant for: GUI actions and components

The Object Repository Manager toolbar contains the following buttons:

Button	Description
	New. Enables you to create a new shared object repository. For details, see "Create a new shared object repository" on page 1291.
	Open. Enables you to open a shared object repository from the file system or from ALM. For details, see "Open a shared object repository" on page 1292.
	Save. Enables you to save the active shared object repository to the file system or to ALM. For details, see "Save a shared object repository" on page 1292.

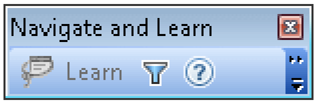
Button	Description
✎	**Enable Editing.** Enables you to edit the active shared object repository, by making the shared object repository editable. For details, see "Enable editing for a shared object repository" on page 1292.
↶	**Undo.** Enables you to undo the previous operation performed in the active shared object repository. You do this in the same way as in a local object repository. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
↷	**Redo.** Enables you to redo the operation that was previously undone in the active shared object repository. You do this in the same way as in a local object repository. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
✂	**Cut.** Enables you to cut the selected item or object in the active shared object repository. You do this in the same way as in a local object repository. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
⧉	**Copy.** Enables you to copy the selected item or object to the Clipboard in the active shared object repository. You do this in the same way as in a local object repository. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
📋	**Paste.** Enables you to paste the data from the Clipboard to the active shared object repository. You do this in the same way as in a local object repository. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
✕	**Delete.** Enables you to delete the selected item or object in the active shared object repository. You do this in the same way as in a local object repository. For details, see "How to Copy, Paste, Move, or Delete Objects in the Object Repository" on page 1217.
🔍	**Find and Replace.** Enables you to find an object, property, or property value in the active shared object repository. You can also find and replace specified property values. You do this in the same way as in a local object repository. For details, see "Find and Replace Dialog Box (Object Repository)" on page 1245.
✚	**Add Objects.** Enables you to add objects to the active shared object repository. You do this in the same way as in a local object repository. For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199.
⟳	**Update from Application.** Enables you to update identification properties in the active shared object repository according to the actual properties of the object in your application. You do this in the same way as in a local object repository. For details, see "Updating Identification Properties from an Object in Your Application" on page 1206.
⊞	**Add Insight Object.** Enables you to add an Insight object to the active shared object repository. You do this in the same way as in a local object repository. For details, see "Adding Insight Test Objects" on page 1200.

Button	Description
✳	**Define new Test Object.** Enables you to define a test object that does not yet exist in your application and add it to the active shared object repository. You do this in the same way as in a local object repository. For details, see "Define New Test Object Dialog Box" on page 1240.
⬡	**Highlight in Application.** Enables you to select an object in the active shared object repository and highlight it in your application. You do this in the same way as in a local object repository. For details, see "How to Locate an Object in an Object Repository" on page 1219.
⌖	**Locate in Repository.** Enables you to select an object in your application and highlight it in the active shared object repository. You do this in the same way as in a local object repository. For details, see "Locating a Test Object in the Object Repository" on page 1204.
🔗	**ALM Connection.** Enables you to connect to ALM to work with object repository files stored in an ALM project. You can connect to ALM from the main UFT window or from the Object Repository Manager. For details, see "ALM Connection Dialog Box" on page 737.
🕵	**Object Spy.** Opens the "Object Spy Dialog Box " (described on page 1189), enabling you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and operations of the test object that UFT uses to represent that object. For details, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 1184.
<P>	**Manage Repository Parameters.** Enables you to add, edit, and delete repository parameters in the active shared object repository. For details, see "Manage Repository Parameters Dialog Box" on page 1306.

Navigate and Learn Toolbar

Relevant for: GUI actions and components

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



To access	In the Object Repository Manager, do one of the following: - Select **Object > Navigate and Learn**. - Press F6.

Important information	- Each time you select a window to learn, the selected window and its descendant objects are added to the active shared object repository according to a predefined object filter. You can change the object filter definitions at any time to meet your requirements. The object filter is used for both the **Navigate and Learn** option and the **Add Objects** option. The settings you define are used in both places when UFT learns objects. For details on modifying the filter definitions, see "Define Object Filter Dialog Box" on page 1242. - The Navigate and Learn option is not supported for environments with mixed hierarchies (object hierarchies that include objects from different environments), for example, Browser("Homepage").Page ("Welcome").AcxButton("Save") or Dialog("Edit").AcxEdit("MyEdit"). To add objects within mixed hierarchies, use other options, as described in "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 1199. - The **Navigate and Learn** option does not learn the following types of objects: ▪ Minimized windows ▪ Insight test objects ▪ Objects in Google Chrome Frame
Relevant tasks	"How to Create and Manage Shared Object Repositories" on page 1291
See also	"Object Repository Manager Main Window" on page 1297

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

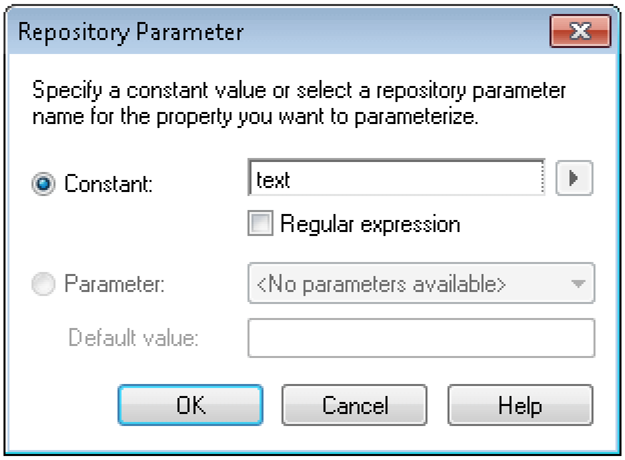
UI Elements	Description
Learn	Adds the active (in focus) parent object and its descendants to the shared object repository, according to the defined filter. **Note:** This button is disabled if there is no recognized active parent object (for example, Browser, Dialog, Window). **Keyboard shortcut:**ENTER

UI Elements	Description
▽	**Define Object Filter.** Opens the "Define Object Filter Dialog Box" (described on page 1242), enabling you to set filter definitions for objects learned. The current filter definitions are displayed in the button tooltip (in parentheses after the button name). **Note:** If this is the first time you are adding objects to the shared object repository, you may want to change the filter definitions before you continue. **Keyboard shortcut:** CTRL+F
<keyboard shortcuts>	• **Help.** F1 • **Return to Object Repository Manager.** ESC

Repository Parameter Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to specify or modify property values for test objects in the shared object repository. You can specify a value using a constant value (either a simple value or a constant value that includes regular expressions), or you can parameterize a value using a repository parameter.



Specify a constant value or select a repository parameter name for the property you want to parameterize.

● Constant: text

□ Regular expression

○ Parameter: <No parameters available>

Default value:

To access	In the "Object Repository Manager Main Window" (described on page 1297), do the following: 1. Select the test object whose property value you want to specify. 2. In the **Test object details** area, click in the **Value** cell for the required property.

Important information	You can also specify or modify values for properties of a checkpoint or output object, directly in the object details area of the Object Repository Manager. For details, see "Object Repository Manager Main Window" on page 1297.
Relevant tasks	"How to Manage Objects in Shared Object Repositories" on page 1294
See also	• "Working with Repository Parameters" on page 1287 • "Manage Repository Parameters Dialog Box" on the next page

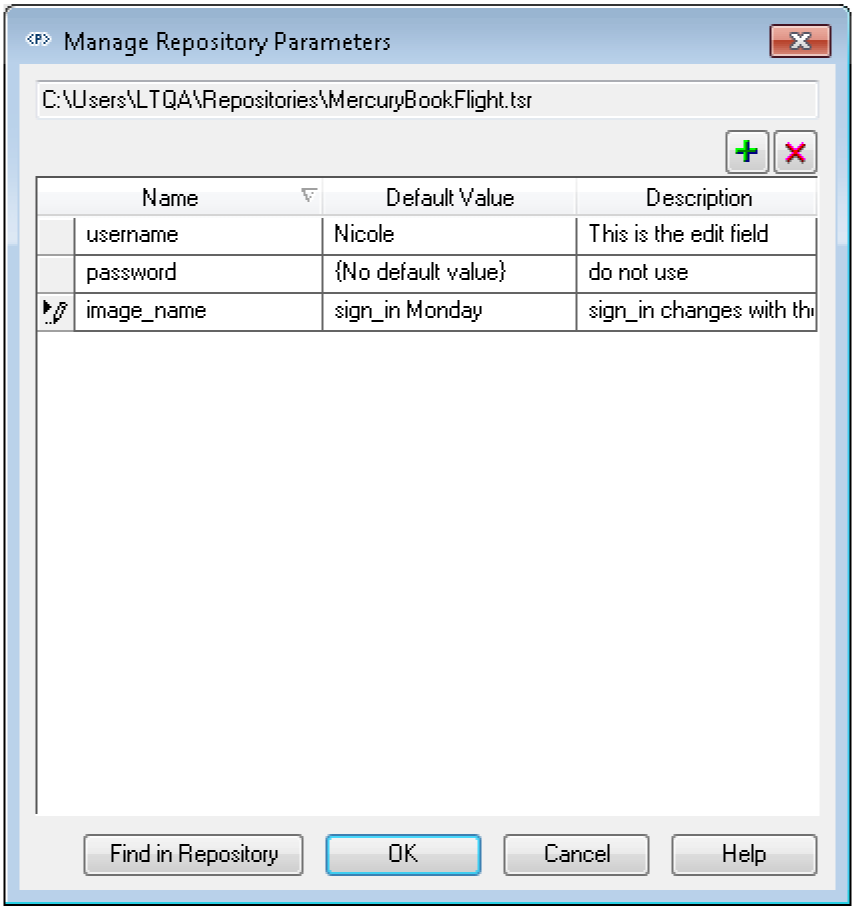
User interface elements are described below:

UI Elements	Description
Constant	The constant value of the object property. **Note:** You can also enter a constant value directly in the **Value** cell of the **Test object details** area.
Regular expression	Enables you to specify a regular expression for the constant value. For details on regular expressions, see "Regular Expressions Overview" on page 318 and "Regular Expression Characters and Usage Options" on page 352.
Parameter	Enables you to select a repository parameter from the list of defined parameters.
Default value	The default value for the parameter, where applicable.

Manage Repository Parameters Dialog Box

Relevant for: GUI tests and components

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



C:\Users\LTQA\Repositories\MercuryBookFlight.tsr

Name	Default Value	Description
username	Nicole	This is the edit field
password	{No default value}	do not use
image_name	sign_in Monday	sign_in changes with th

To access	In the "Object Repository Manager Main Window" (described on page 1297), do one of the following: - Select **Tools > Manage Repository Parameters**. - Click the **Manage Repository Parameters** button .
Relevant tasks	"How to Manage Objects in Shared Object Repositories" on page 1294
See also	- "Working with Repository Parameters" on page 1287. - "Add Repository Parameter Dialog Box" on page 1308 - "Repository Parameter Dialog Box" on page 1304

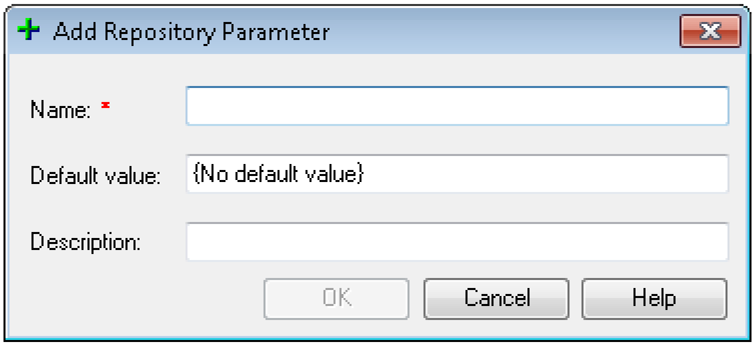
User interface elements are described below:

Option	Description
<Repository>	Displays the name and path of the shared object repository whose repository parameters you are managing.
➕	**Add Repository Parameter.** Enables you to add a new repository parameter. For details, see "Add Repository Parameter Dialog Box" on the next page.
✕	**Delete Repository Parameter.** Enables you to delete the currently selected repository parameters.
Parameter list (Name, Default Value, and Description)	Displays the list of repository parameters currently defined in this shared object repository. You can modify a parameter's default value and description directly in the parameter list. For considerations, see "Working with Repository Parameters" on page 1287.
✎	**Clear Default Value**. Enables you to remove the default value of the parameter. If you remove the default value, the text **{No Default Value}** is displayed in the cell. **Note:** If you delete the text manually, it does not remove the default value. It creates a default value of an empty string. You must click the **Clear Default Value** button if you want to remove the default value.
Find in Repository	Searches for and highlights the first test object in the shared object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so on.

Add Repository Parameter Dialog Box

Relevant for: GUI tests and components

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



Name: *

Default value: {No default value}

Description:

To access	In the "Manage Repository Parameters Dialog Box" (described on page 1306), click the **Add Repository Parameter** button ✚.
Relevant tasks	"How to Manage Objects in Shared Object Repositories" on page 1294
See also	• "Working with Repository Parameters" on page 1287 • "Manage Repository Parameters Dialog Box" on page 1306 • "Repository Parameter Dialog Box" on page 1304

User interface elements are described below:

UI Elements	Description
Name	A meaningful name for the parameter. For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269.

UI Elements	Description
Default value	The default value to be used for the repository parameter. This value is used if you do not map the repository parameter to a value or parameter type in a test or component that uses this shared object repository. If you do not specify a default value, the repository parameter will appear as unmapped in any tests or components that use this shared object repository. **Note:** If you specify a default value, you can later remove it by clicking in the **Default Value** cell of the relevant parameter in the "Manage Repository Parameters Dialog Box" (described on page 1306) and then clicking the **Clear Default Value** button. The text **{No Default Value}** is displayed in the cell.
Description	A description of the repository parameter. The description will help you identify the parameter when mapping repository parameters in a test or component.

Chapter 46: Configuring Object Identification

Relevant for: GUI tests and components

This chapter includes:

Concepts

Object Identification Configuration - Overview

Relevant for: GUI tests and components

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

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

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

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

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

Note:

- If the combination of all defined mandatory and assistive properties is not sufficient to create a unique test object description, UFT also learns the value for the selected ordinal identifier. For details, see "Ordinal Identifiers" on the next page.

- If a specific test object relies mainly on ordinal identifiers, you can also define visual relation identifiers for that test object, to help improve identification reliability for that object. For details, see "Visual Relation Identifiers" on page 1209.

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

You use the "Object Identification Dialog Box" (described on page 1325) to configure the mandatory, assistive, and ordinal identifier properties that UFT uses to learn descriptions of the objects in your application, and to enable and configure the Smart Identification mechanism.

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

Mandatory and Assistive Properties

Relevant for: GUI tests and components

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

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

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

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

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

Ordinal Identifiers

Relevant for: GUI tests and components

In addition to learning the mandatory and assistive properties specified in the "Object Identification Dialog Box" (described on page 1325), UFT can also learn a backup ordinal identifier for each test object. The **ordinal identifier** assigns the object a numerical value that indicates its order relative to other objects with an otherwise identical description (objects that have the same values for all properties specified in the mandatory and assistive property lists). This ordered value enables UFT to create a unique description when the mandatory and assistive properties are not sufficient to do so.

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

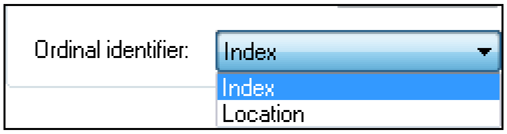
In addition, even if UFT learns an ordinal identifier, it will use the identifier during the run session only if:

- The learned description and the Smart Identification mechanism are not sufficient to identify the object in your application.
- A visual relation identifier is not defined for the test object. For details, see "Visual Relation Identifiers" on page 1209.

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

- **Index.** Indicates the order in which the object appears in the application code relative to other objects with an otherwise identical description. For details, see "The Index Ordinal Identifier" on the next page.
- **Location.** Indicates the order in which the object appears within the parent window, frame, or dialog box relative to other objects with an otherwise identical description. For details, see "The Location Ordinal Identifier" on the next page.
- **CreationTime.** (Browser object only.) Indicates the order in which the browser was opened relative to other open browsers with an otherwise identical description. For details, see "The CreationTime Ordinal Identifier" on page 1316.

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



Tip: When learning an object, if UFT successfully creates a unique test object description using the mandatory and assistive properties, it does not learn an ordinal identifier value. You can add an ordinal identifier to an object's identification properties at a later time using the "Ordinal Identifier Dialog Box", available from the Object Properties or Object Repository window. For details, see "Managing Test Objects in Object Repositories" on page 1198.

This section also includes:

- "The Index Ordinal Identifier" below
- "The Location Ordinal Identifier" below
- "The CreationTime Ordinal Identifier" on page 1316

The Index Ordinal Identifier

Relevant for: GUI tests and components

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

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

For example, suppose a page contains the following objects:

- An image with the name Apple
- An image with the name UserName
- A WebEdit object with the name UserName
- An image with the name Password
- A WebEdit object with the name Password

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

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

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

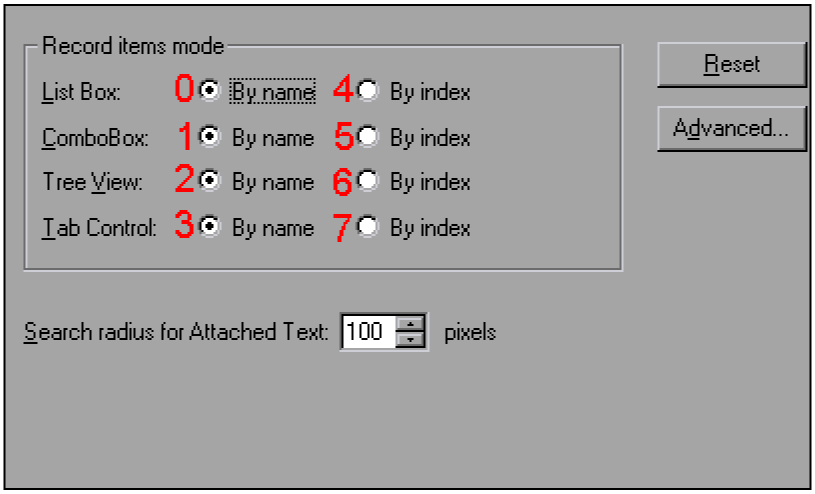
```
WebElement("Name:=UserName", "Index:=0")
```

The Location Ordinal Identifier

Relevant for: GUI tests and components

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

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



Record items mode

List Box: 0 ● By name 4 ○ By index

ComboBox: 1 ● By name 5 ○ By index

Tree View: 2 ● By name 6 ○ By index

Tab Control: 3 ● By name 7 ○ By index

Search radius for Attached Text: 100 pixels

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

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

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

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

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

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

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

The CreationTime Ordinal Identifier

Relevant for: GUI tests and components

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

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

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

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

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

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

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

Smart Identification

Relevant for: GUI tests and components

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

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

The "Smart Identification Properties Dialog Box" (described on page 1331) enables you to create and modify the Smart Identification definition that UFT uses for a selected test object class. Configuring Smart Identification properties enables you to help UFT identify objects in your application, even if some of the properties in the object's learned description have changed.

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

The Smart Identification mechanism uses two types of properties:

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

Smart identification is not relevant for Insight test objects.

This section also includes:

When to Use Smart Identification

Relevant for: GUI tests and components

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

Even if you choose to learn Smart Identification properties for an object, you can disable use of the Smart Identification mechanism for a specific object in the Object Properties or Object Repository window. For tests, you can also disable use of the mechanism for an entire test in the Run pane of the Test Settings dialog box. For details, see "Managing Test Objects in Object Repositories" on page 1198, and, for tests, "Run Pane (Test Settings Dialog Box)" on page 601.

However, if you do not learn Smart Identification properties, you cannot enable the Smart Identification mechanism for an object later. For details on learning Smart Identification properties, see "Smart Identification Properties Dialog Box" on page 1331.

The Smart Identification Process

Relevant for: GUI tests and components

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

1. UFT "forgets" the learned test object description and creates a new **object candidate** list containing the objects (within the object's parent object) that match all of the properties defined in the Base Filter Properties list.

2. UFT filters out any object in the object candidate list that does not match the first property listed in the Optional Filter Properties list. The remaining objects become the new object candidate list.

3. UFT evaluates the new object candidate list:

   - If the new object candidate list still has more than one object, UFT uses the new (smaller) object candidate list to repeat the filter for the next optional filter property in the list.

   - If the new object candidate list is empty, UFT ignores this optional filter property, returns to the previous object candidate list, and repeats the filter for the next optional filter property in the list.

   - If the object candidate list contains exactly one object, then UFT concludes that it has identified the object and performs the statement containing the object.

4. UFT continues the filtering process described above until it either identifies one object, or runs out of optional filter properties to use.

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

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

Smart Identification Information in the Run Results

Relevant for: GUI tests and components

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

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

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

For more details, see Smart Identification in the Run Results (described in the *HP Run Results Viewer User Guide*).

How UFT Uses Smart Identification - Use-Case Scenario

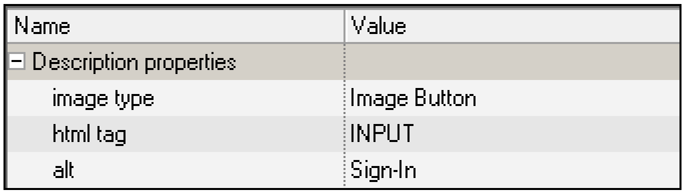
Relevant for: GUI tests and components

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

Suppose you have the following statement in your test or component:

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

When you created your test or component, UFT learned the following object description for the Login image:



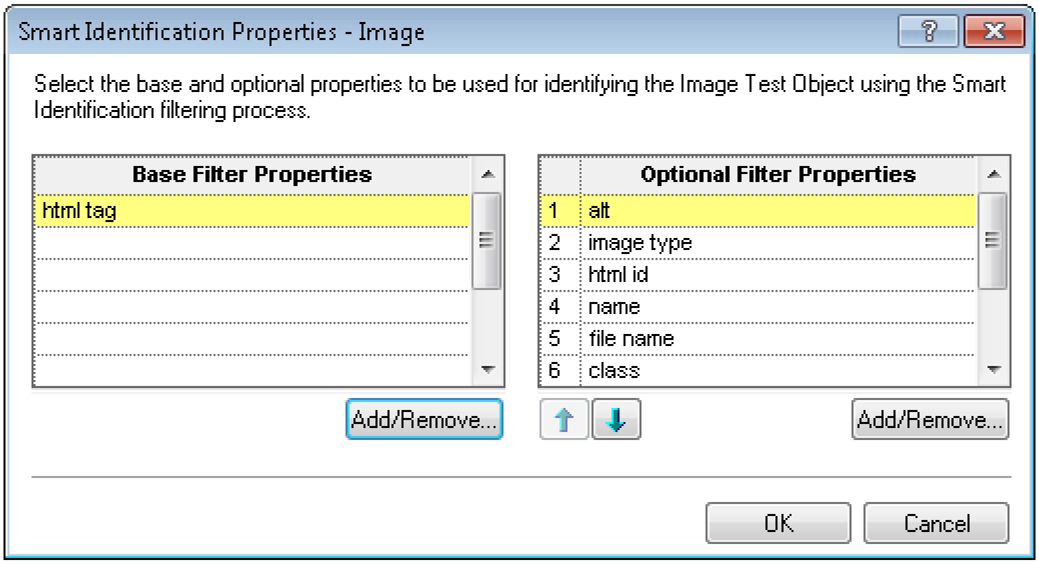
Name	Value
⊟ Description properties	
image type	Image Button
html tag	INPUT
alt	Sign-In

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

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

The following explanation describes the process that UFT uses to find the Login object using Smart Identification:

1. According to the Smart Identification definition you have for Web image objects, UFT learned the values of the following properties when it learned the Login image:



Smart Identification Properties - Image
Select the base and optional properties to be used for identifying the Image Test Object using the Smart Identification filtering process.
Base Filter Properties
html tag
Optional Filter Properties
1 alt
2 image type
3 html id
4 name
5 file name
6 class
Add/Remove...
Add/Remove...
OK
Cancel

The learned values are as follows:

Base Filter Properties:

Property	Value
html tag	INPUT

Optional Filter Properties:

Property	Value
alt	Login
name	login
file name	login.gif
class	<null>
visible	1

2. UFT begins the Smart Identification process by identifying the five objects on the Mercury Tours page that match the base filter properties definition (**html tag** = INPUT). UFT considers these to be the object candidates and begins checking the object candidates against the **Optional Filter Properties** list.

3. UFT checks the **alt** property of each of the object candidates, but none have the **alt** value: Login, so UFT ignores this property and moves on to the next one.

4. UFT checks the name property of each of the object candidates, and finds that two of the objects (both the basic and VIP Login buttons) have the name: login. UFT filters out the other

three objects from the list, and these two login buttons become the new object candidates.

5. UFT checks the file name property of the two remaining object candidates. Only one of them has the file name login.gif, so UFT correctly concludes that it has found the Login button and clicks it.

Test Object Mapping for Unidentified or Custom Classes

Relevant for: GUI tests and components

The "Object Mapping Dialog Box" (described on page 1330) enables you to map an object of an unidentified or custom class to a standard Windows class. For example, if your application has a button that cannot be identified, this button is learned as a generic WinObject.

You can teach UFT to identify your object as if it belonged to a standard Windows button class. Then, when you click the button while recording, UFT records the operation in the same way as a click on a standard Windows button.

When you map an unidentified or custom object to a standard object, your object is added to the list of standard Windows test object classes as a user-defined test object class. You can configure the object identification settings for a user-defined test object class just as you would any other test object class.

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

Tasks

How to Configure Object Identification for a Test Object Class

Relevant for: GUI tests and components

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

Repeat the task for all relevant test object classes.

This task includes the following steps:

- "Prerequisites " below
- "Set the properties that UFT learns for identifying an object in this test object class" below
- "Set the properties that UFT uses for Smart Identification" below

Prerequisites

Determine what you would like to change about how UFT identifies objects of various test object classes within your application. For details, see "Object Identification Configuration - Overview" on page 1311.

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

In the "Object Identification Dialog Box" (described on page 1325), set the properties that are learned for the test object description:

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

Set the properties that UFT uses for Smart Identification

1. In the "Object Identification Dialog Box", select the **Enable Smart ID** check box. The **Configure** button is enabled.
2. Click the **Configure** button to open the "Smart Identification Properties Dialog Box" (described on page 1331).

3. Set the base and optional properties.

4. Set the order in which the optional properties are used.

5. If you do not want UFT to learn the Smart Identification properties, clear the **Enable Smart Identification** check box. The configuration is saved, but not used.

   For more details, see "Smart Identification" on page 1316.

How to Manage Identification Properties of a Test Object Class

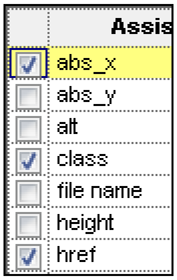
Relevant for: GUI tests and components

The following task describes how to manage the identification properties of a specified test object class in the "Add/Remove Properties Dialog Box" (described on page 1328) and includes the following:

- "Add and remove properties to the object identification property lists" below
- "Add identification properties to the test object class" below

Add and remove properties to the object identification property lists

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



	Assis
☑	abs_x
□	abs_y
□	alt
☑	class
□	file name
□	height
☑	href

Add identification properties to the test object class

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

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

Relevant for: GUI tests and components

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

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

1. Open the "Object Mapping Dialog Box" (described on page 1330) by selecting **Standard Windows** in the **Environment** box in the Object Identification dialog box and then clicking the **User-Defined** button.

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

   For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.

3. In the **Map to** box, select the standard object class to which you want to map your user-defined test object class and click **Add**. The class name and mapping is added to the object mapping list.

4. Click **OK**. The Object Mapping dialog box closes and your object is added to the list of standard Windows test object classes as a user-defined test object class. Note that your object has an icon with a red U in the lower-right corner, identifying it as a user-defined class.

5. Configure the object identification settings for your user defined test object class just as you would any other test object class. For details, see "How to Configure Object Identification for a Test Object Class" on page 1322.

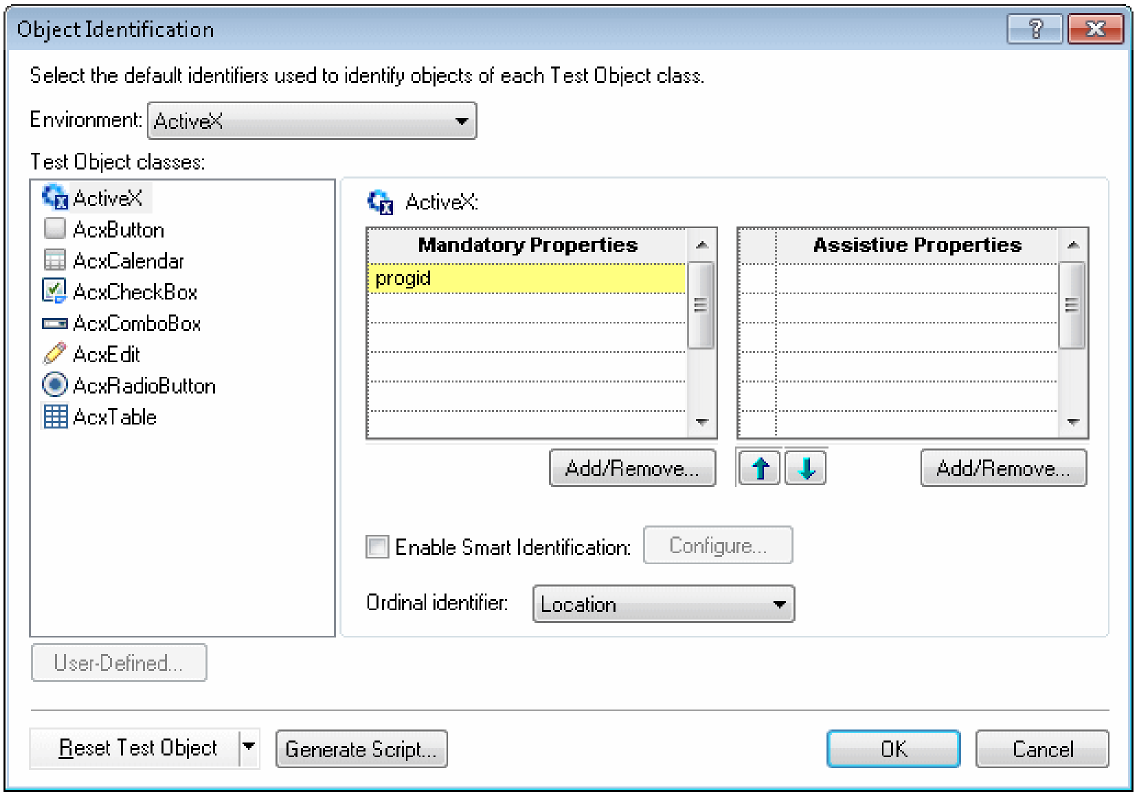
> **Caution:** If you click the down arrow on the **Reset Test Object** button and select **Reset Environment**, when **Standard Windows** is selected in the **Environment** box, all of the user-defined test object classes are deleted.

Reference

Object Identification Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to set mandatory and assistive properties, to select the ordinal identifier, and to specify whether you want to enable the Smart Identification mechanism for each test object class.



Object Identification

Select the default identifiers used to identify objects of each Test Object class.

Environment: ActiveX

Test Object classes:

- ActiveX
- AcxButton
- AcxCalendar
- AcxCheckBox
- AcxComboBox
- AcxEdit
- AcxRadioButton
- AcxTable

ActiveX:

Mandatory Properties
progid

Assistive Properties

□ Enable Smart Identification:

Ordinal identifier: Location

To access	1. 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. 2. Select **Tools > Object Identification**.

Important information	- Only currently loaded environments are listed in the **Environments** box. - You cannot include the same property in both the mandatory and assistive property lists. - Any changes you make in the Object Identification dialog box have no effect on objects already added to existing object repositories. To update the test objects in the object repository, run the test or component in Update Run Mode using the **Update test object descriptions** option. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107. - The learned and Smart Identification properties of certain classes cannot be configured, for example, VbLabel objects. Such classes are therefore not included in the **Test Object classes** list for the selected environment. - Insight test objects are not available in this dialog box.
Relevant tasks	"How to Configure Object Identification for a Test Object Class" on page 1322
See also	"Object Identification Configuration - Overview" on page 1311

User interface elements are described below:

UI Elements	Description
Environment	The list of environments, according to the loaded add-ins. This list may also include additional environments for which you or a third party developed support using UFT Add-in Extensibility. For details on loading UFT add-ins or on UFT Add-in Extensibility, see the relevant sections in the *HP Unified Functional Testing Add-ins Guide*.
Test Object classes	The list of the test object classes associated with the selected environment. **Note:** In **Standard Windows**, the user-defined classes are displayed at the bottom of the list.
Mandatory Properties	The list of properties that UFT always learns as part of the description for test objects of the selected class.
Assistive Properties	The list of additional properties that UFT can learn for a test object of the selected class to create a unique test object description.

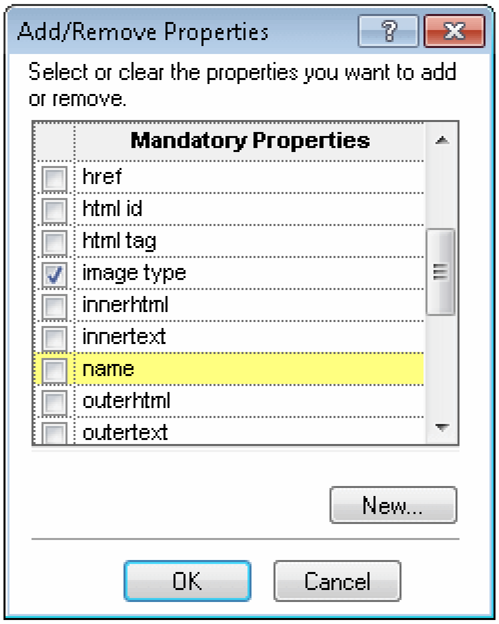
UI Elements	Description
↑ ↓	**Up and Down arrows.** Enable you to set the preferred order for the **Assistive Properties** list. When UFT learns an object, and assistive properties are necessary to create a unique object description, UFT adds the assistive properties to the description one at a time until it has enough information to create a unique description, according to the order you set in the **Assistive Properties** list.
Add/Remove	Enables you to add or remove properties for the test object class using the "Add/Remove Properties Dialog Box"(described on page 1328).
Enable Smart Identification	Enables or disables Smart Identification for the selected test object class.
Configure	Opens the "Smart Identification Properties Dialog Box" (described on page 1331), which enables you to specify the properties for UFT to learn as Smart Identification Properties. Enabled only when the **Enable Smart Identification** check box is selected.
Ordinal identifier	The type of ordinal identifier to use when UFT needs to identify an object with an otherwise identical description. The ordinal identifier enables UFT to assign a serial order value to each of the otherwise identical objects. For details, see "Ordinal Identifiers" on page 1312.
User-Defined	Enables you to map an unidentified or custom class to a standard Windows class by opening the "Object Mapping Dialog Box". For details, see "Object Mapping Dialog Box" on page 1330. **Note:** To enable this option, you must select the **Standard Windows** environment from the **Environment** list.

UI Elements	Description
Reset Test Object	The down arrow enables you to select one of the following options: • **Reset Test Object.** Resets the selected test object to the system default. (Default selection) • **Reset Environment.** Resets the settings for all the test objects in the current environment to the system default. • **Reset All.** Resets the settings for all currently loaded environments to the system default. You can restore the default settings for object identification and the Smart Identification property settings for all loaded environments, for the current environment only, or for a selected test object, using the **Reset Test Object** option in the "Object Identification Dialog Box". For details, see "Object Identification Dialog Box" on page 1325. Only built-in object properties can be reset. When you reset the settings for the standard Windows environment, user-defined objects are also deleted. For details on user-defined objects, see "Test Object Mapping for Unidentified or Custom Classes" on page 1321.
Generate Script	Enables you to generate an automation script that sets the current object identification settings. When you click the **Generate Script** button, a Save As dialog box opens, enabling you to specify the name and file system location to store the generated file. You can use some or all of the script lines from this generated script in an automation script. This can be useful, for example, if you to set the same object identification settings on multiple UFT computers. For more details, see "UFT Automation Scripts" on page 1155 and the *HP UFT Automation Object Model Reference*.

Add/Remove Properties Dialog Box

Relevant for: GUI tests and components

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



To access	You can access this dialog box by clicking **Add/Remove** in the following dialog boxes: • "Object Identification Dialog Box" (described on page 1325) • "Smart Identification Properties Dialog Box" (described on page 1331)
Important information	• You cannot add the same property to both the mandatory and assistive property lists. • You cannot add the same property to both the base and optional filter property lists.
Relevant tasks	"How to Configure Object Identification for a Test Object Class" on page 1322
See also	• "Object Identification Configuration - Overview" on page 1311 • "Smart Identification" on page 1316

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

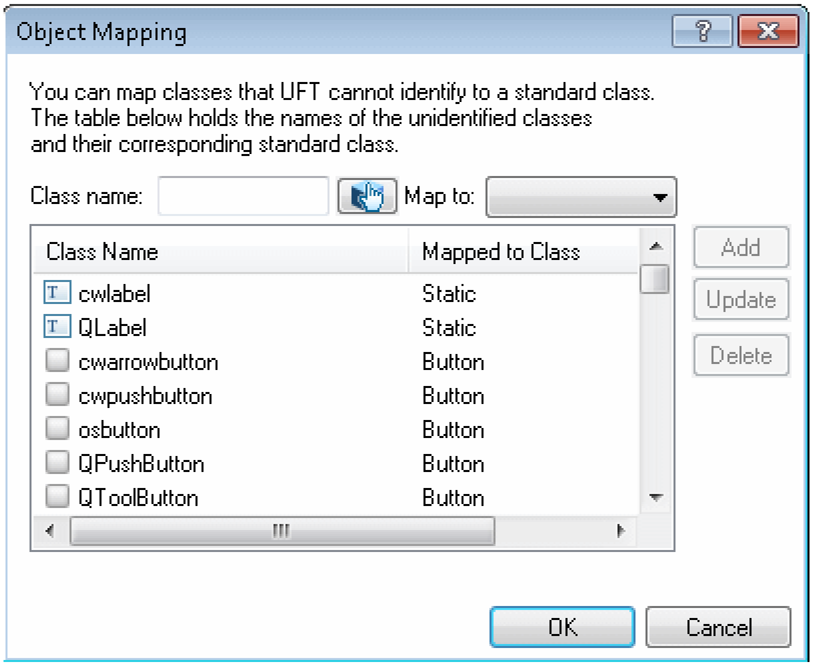
UI Elements	Description
<Properties> list	The list of available identification properties for the selected test object class. For details, see "Add and remove properties to the object identification property lists" on page 1323.

UI Elements	Description
New	Enables you to add new identification properties to the available properties list. Enter a valid property in the format attribute/<*PropertyName*> (for Web objects) or <*PropertyName*> (for other environment objects). UFT retrieves the property values from the object's native properties. Therefore, the name of a new identification property must be identical to the name of a native property. (In most environments, you can use the Object Spy to view the list of available native properties.) If you add a property to a Java test object, see the section on Java Add-in considerations in the *HP Unified Functional Testing Add-ins Guide*. **Note:** New properties are displayed in the Object Spy but are not available for checkpoints on objects of the selected class. For more details, see "Add identification properties to the test object class" on page 1323.

Object Mapping Dialog Box

Relevant for: GUI tests and components

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



You can map classes that UFT cannot identify to a standard class. The table below holds the names of the unidentified classes and their corresponding standard class.

Class name:
Map to:

Class Name	Mapped to Class
cwlabel	Static
QLabel	Static
cwarrowbutton	Button
cwpushbutton	Button
osbutton	Button
QPushButton	Button
QToolButton	Button

To access	In the "Object Identification Dialog Box" (described on page 1325), select the **Standard Windows** environment, and click the **User Defined** button.
Relevant tasks	"How to Map an Unidentified or Custom Class to a Standard Windows Class" on page 1324
See also	"Test Object Mapping for Unidentified or Custom Classes" on page 1321

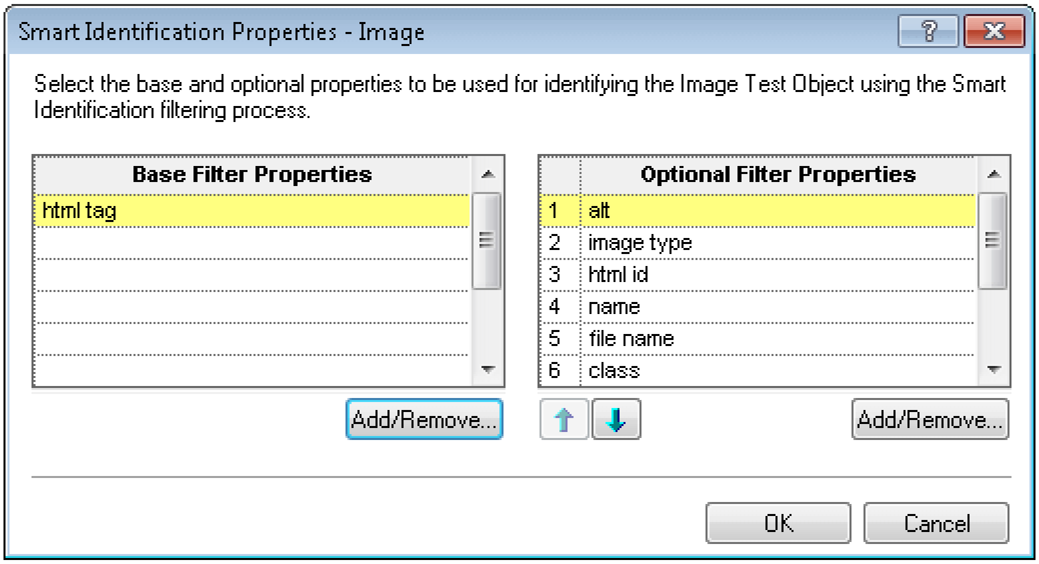
User interface elements are described below:

UI Elements	Description
Class name	The name of the user-defined object selected by using the **Pointing Hand**.
👆	**Pointing Hand.** Enables you to add the object whose class you want to as a user-defined class by minimizing the UFT window and navigating to the object. The name of the user-defined object is displayed in the **Class name** box. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.
Map to	The list of available standard Windows test object classes to map to the user-defined object.
Add	Enables you to add the selected class name and mapping in the object mapping list with values set in the **Class Name** and **Map to** fields.
Update	Enables you to update the selected class name and mapping in the object mapping list with new values set in the **Class Name** and **Map to** fields.
Delete	Enables you to delete the selected class name and mapping from the object mapping list.

Smart Identification Properties Dialog Box

Relevant for: GUI tests and components

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



Smart Identification Properties - Image

Select the base and optional properties to be used for identifying the Image Test Object using the Smart Identification filtering process.

Base Filter Properties
html tag

	Optional Filter Properties
1	alt
2	image type
3	html id
4	name
5	file name
6	class

To access	In the "Object Identification Dialog Box" (described on page 1325), select an environment and a test object class, select **Enable Smart Identification**, and click **Configure**.
Important information	By default, some test object classes already have Smart Identification configurations and others do not. Those with default configurations also have the **Enable Smart Identification** check box selected by default in the "Object Identification Dialog Box" (described on page 1325).
See also	"Smart Identification" on page 1316

User interface elements are described below:

UI Elements	Description
Base Filter Properties	The list of properties that UFT learns as base filter properties for this test object class. The Smart Identification mechanism uses these properties to create a list of possible candidate objects.
Optional Filter Properties	The list of properties that UFT learns as optional filter properties for this test object class. The Smart Identification mechanism uses these properties in the specified order to narrow down the object candidate list to one object.
↑ ↓	**Up and Down arrows.** Enable you to set the preferred order for the **Optional Filter Properties** list. When UFT uses Smart Identification, it creates a list of possible candidate objects according the **Base Filter Properties**, and then checks the values of the **Optional Filter Properties** one by one according to the order you set, until it narrows down the candidate list to one object.
Add/Remove	Opens the "Add/Remove Properties Dialog Box", enabling you to modify the list of properties learned for Smart Identification for this test object class. For details, see "Add/Remove Properties Dialog Box" on page 1328.

Chapter 47: Object Repository Comparison Tool

Relevant for: GUI tests and components

This chapter includes:

Concepts

Object Repository Comparison Tool Overview

Relevant for: GUI tests and components

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

Differences between objects in the two object repository files are identified according to default rules. During the comparison process, the object repository files remain unchanged. For details about the types of comparisons identified by the Object Repository Comparison Tool, see "Understanding Object Differences" on page 1347.

After the comparison process, the Comparison Tool provides a graphic presentation of the objects in the object repositories, which are shown as nodes in a hierarchy. Objects that have differences, as well as unique objects that are included in one object repository only, can be identified according to a color configuration that you can specify. Objects that are included in one object repository only are indicated in the other object repository by the text "Does not exist". You can also view the properties and values of each object that you select in either object repository.

You can use the information displayed by the Object Repository Comparison Tool when managing or merging object repositories. For details, see "Object Repository Merge Tool" on page 1351.

This section also includes:

- "When to Use the Object Repository Comparison Tool" below
- "Understanding the Repository Panes" below

When to Use the Object Repository Comparison Tool

In addition to this tool, you can perform merge and comparison operations using the Asset Comparison Tool and the Object Repository Merge Tool. Consider the following when selecting which tool to use:

- The Object Repository Comparison Tool is designed for comparing repositories that are different, but have a set of overlapping objects. This tool is useful if you want to decide whether to merge two repositories without performing the actual merge and addressing object conflicts in the Object Repository Merge Tool. For details, see "How to Compare Two Object Repositories" on page 1336 and "Object Repository Merge Tool" on page 1351.
- The Asset Comparison Tool can also compare two repositories, but it is designed for comparing different versions of the same repository to identify changes between versions. For details, see "Viewing and Comparing Versions of UFT Assets" on page 770.

Understanding the Repository Panes

The object repository panes of the Object Repository Comparison Tool display the hierarchies of the objects, and their properties and values, in the object repository files that you are comparing.

The file path is shown above each object hierarchy.

To make it easier to see the status of an object at a glance, the text and background of object names in the object repositories are displayed using different colors, according to the type of comparison found.

The Object Repository Comparison Tool enables you to navigate the two object repositories independently. You can also resize the various panes to display only some of the objects contained in the object repositories. When using large object repositories, this can result in the various panes displaying different areas of the object repository hierarchies, making it difficult to locate and track specific objects affected by the comparison process. You can synchronize the object repositories to display the same object in both views.

Tasks

How to Compare Two Object Repositories

Relevant for: GUI tests and components

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

This task includes the following steps:

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

1. Prerequisites

- Determine the shared object repositories you want to compare. Generally, shared object repositories should contain objects from the same application, and may have differences in objects or test object descriptions because the repositories were created at different times or under different circumstances.
- Make sure that a GUI test or component is currently open.
- Make sure that the Object Repository Manager window is open. For details on working with the Object Repository Manager, see "Object Repository Manager Main Window" on page 1297.
- Make sure the color settings are configured to match your needs. For details, see "Color Settings Dialog Box (Object Repository Comparison Tool) " on page 1348.

2. Select the Shared Object Repositories to compare

a. In the Object Repository Manager window, select **Tools > Object Repository Comparison Tool** to open the Object Repository Comparison Tool. The "New Comparison Dialog Box" (described on page 1349) opens.

b. Specify the two object repository files you want to compare.

3. Analyze the initial comparison results

After the comparison is complete, you can view the results summary in the "Comparison Statistics Dialog Box" (described on page 1346).

4. **Analyze the detailed comparison results**

   Review and analyze the comparisons between the repositories in the "Object Repository Comparison Tool Main Window" (described on page 1346).

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

   - Synchronize the object repositories to display the same object in both views by clicking the **Synchronized Nodes** button using the "Menu Commands and Toolbar Buttons" (described on page 1339).

   - Filter the objects and show only the objects that you want to view using the "Filter Dialog Box (Object Repository Comparison Tool)" (described on page 1342).

   - Locate one or more objects in a selected object repository whose name contains a specified string using the "Find Dialog Box (Object Repository Comparison Tool)" (described on page 1344).

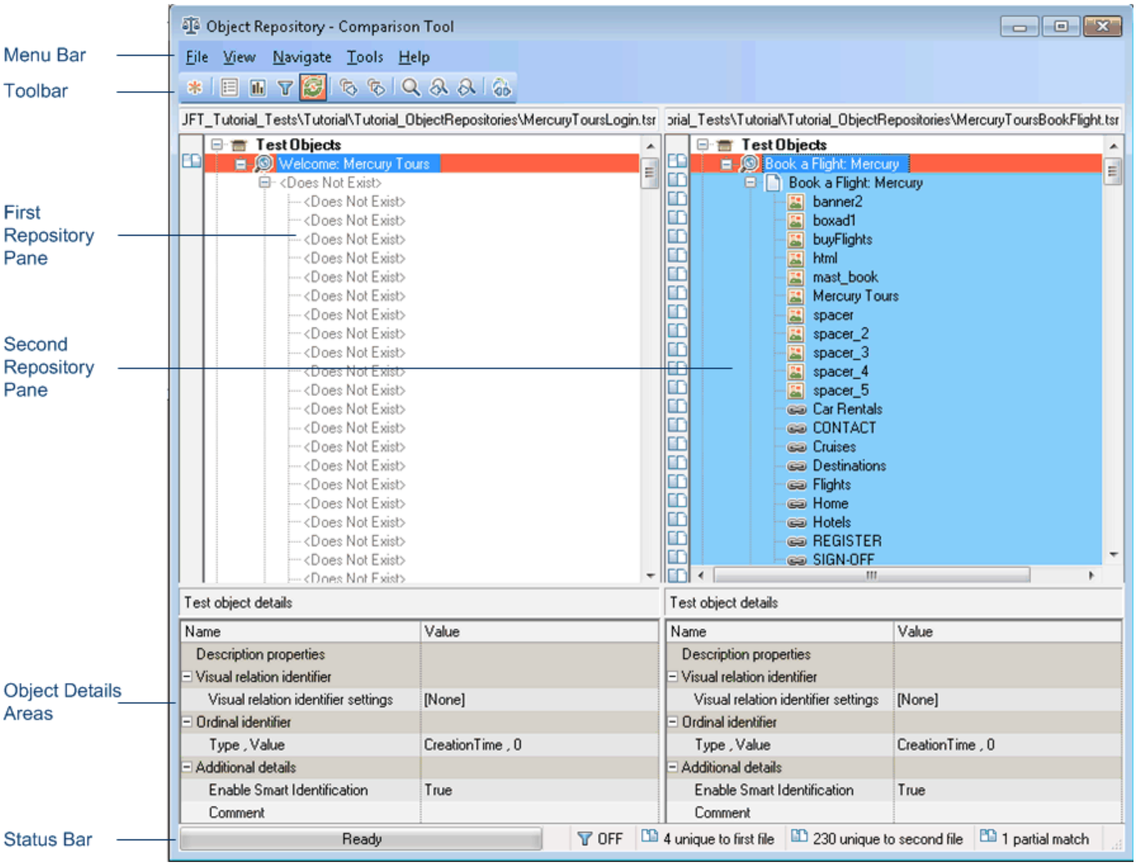
   - Adjust the colors of text and background of object names, and empty nodes representing objects that exist in the other object repository only, using the "Color Settings Dialog Box (Object Repository Comparison Tool) " (described on page 1348).

Reference

Object Repository Comparison Tool Main Window

Relevant for: GUI tests and components

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



Name	Value
Description properties	
Visual relation identifier	
Visual relation identifier settings	[None]
Ordinal identifier	
Type , Value	CreationTime , 0
Additional details	
Enable Smart Identification	True
Comment	

Name	Value
Description properties	
Visual relation identifier	
Visual relation identifier settings	[None]
Ordinal identifier	
Type , Value	CreationTime , 0
Additional details	
Enable Smart Identification	True
Comment	

To access	In the **Object Repository Manager**, select **Tools > Object Repository Comparison Tool**.
Important information	- You cannot work with the Object Repository Manager or the Object Repository Merge Tool while the Object Repository Comparison Tool is open. - The Object Repository Comparison Tool gives precedence to matching test object descriptions over the matching of test object names.
Relevant tasks	"How to Compare Two Object Repositories" on page 1336

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

- "Menu Commands and Toolbar Buttons" below
- "Repository Panes" on the next page
- "Test Object Details Areas" on page 1341
- "Status Bar" on page 1341

Menu Commands and Toolbar Buttons

File Menu

	Command	Shortcut Key	Function
	New Comparison	CTRL+N	Opens the "New Comparison Dialog Box" (described on page 1349), enabling you to specify two object repositories on which to perform a new comparison operation.
	ALM Connection		Enables you to connect the Object Repository Comparison Tool to an ALM project. For details, see "ALM Connection Dialog Box" on page 737.
	Exit		Closes the Object Repository Comparison Tool window.

View Menu

	Command	Function
	Statistics	Opens the "Comparison Statistics Dialog Box" (described on page 1346), which summarizes the comparison between the two repositories, including the number and type of any differences found.
	Collapse All	Collapses the entire hierarchy in both comparison panes. **Tip:** While **Synchronized Nodes** is on, double-clicking an expanded node collapses it in both panes simultaneously.
	Expand All	Expands the entire hierarchy in both comparison panes. **Tip:** Double-clicking a collapsed node expands it in both panes simultaneously.

Navigate Menu

	Command	Shortcut Key	Function
	Next Difference	F4	Finds the next difference between objects in the object repositories.
	Previous Difference	SHIFT +F4	Finds the previous difference between objects in the object repositories.
	Find	CTRL+F	Opens the "Find Dialog Box (Object Repository Comparison Tool)" on page 1344 (described on page 1344).
	Find Next	F3	Finds the next object in the object repositories according to the search specifications in the Find dialog box.
	Find Previous	SHIFT +F3	Finds the previous object in the object repositories according to the search specifications in the Find dialog box.

Tools Menu

	Command	Function
	Synchronized Nodes	Enables you to navigate the two object repository panes simultaneously or independently of one another.
	Filter	Opens the "Filter Dialog Box (Object Repository Comparison Tool)" (described on page 1342), enabling you to specify the comparison types that you want to show.
	Color Settings	Opens the "Color Settings Dialog Box (Object Repository Comparison Tool) " (described on page 1348), enabling you to specify the text and background color of the object names and empty nodes displayed in the comparison panes.

Help Menu

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

Repository Panes

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

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

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

For details on all comparison types, see "Understanding Object Differences" on page 1347.

The object repository panes provide the following functionality:

- While in **Synchronized Nodes** mode, when you select an object in one object repository pane, the corresponding object in the other file hierarchy is located and highlighted. You can press the CTRL button when you select an object to highlight only the selected object without highlighting the corresponding object in the other file.

- When you select an object in an object repository pane, its properties and values are displayed in the respective **Test object details** area at the bottom of the pane.

- When you position your cursor over an icon to the left of an object in an object repository pane, the comparison details are displayed as a tooltip, for example, Partial match, or Unique to second file.

- You can expand or collapse the hierarchy of a parent node by double-clicking the node, or by clicking the expand (**+**) or collapse (**-**) symbol to the left of the node name. You can also expand or collapse the entire hierarchy in the object repository pane by choosing **Collapse All** or **Expand All** from the **View** menu.

- You can jump directly to the next or previous difference in the object repository hierarchy by choosing **Next Difference** or **Previous Difference** from the **Navigate** menu, by clicking the **Next Difference** or **Previous Difference** buttons in the toolbar, or by using keyboard shortcuts. For details about shortcuts, see "Menu Commands and Toolbar Buttons" on page 1339.

- You can drag the edges of the panes to resize them in the Object Repository Comparison Tool window.

Test Object Details Areas

Shows the properties and values of the object selected in an object repository pane. For details, see "Understanding the Repository Panes" on page 1334.

Status Bar

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

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

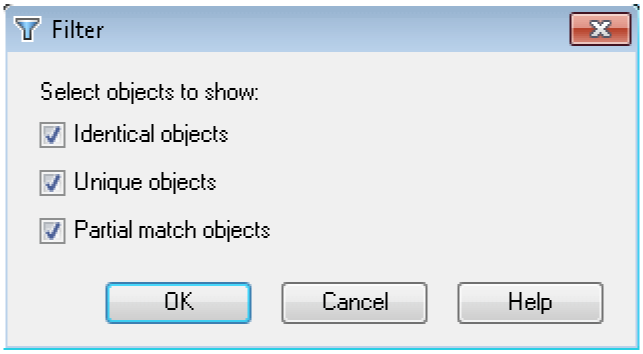
UI Elements	Description
<progress bar>	Displays the comparison process status on the left of the status bar. **Ready** is displayed when the process is complete.
	ALM Connection. Displayed only when UFT is connected to an ALM project.
<filter>	Opens the "Filter Dialog Box (Object Repository Comparison Tool)" below. The icon image indicates the filter status of the repository panes. Possible values: - OFF. Indicates that the object repositories are not filtered and all objects are shown. - ON. Indicates a filter is active and that some objects may have been filtered out of the display.
	The number of objects that are unique to the first file.
	The number of objects that are unique to the second file.
	The number of objects in the first and second file that are not identical, but partially match.

This section also includes:

Filter Dialog Box (Object Repository Comparison Tool)

Relevant for: GUI tests and components

This dialog box enables you to filter objects in the repository panes of the "Object Repository Comparison Tool Main Window". For details, see page 1338.



Filter

Select objects to show:

☑ Identical objects

☑ Unique objects

☑ Partial match objects

To access	In the "Object Repository Comparison Tool Main Window" on page 1338, do one of the following: - Select **Tools > Filter**. - Click the **Filter** button in the toolbar.
Important information	- To view all the objects in both object repositories, select all of the check boxes. - After closing the dialog box, the Filter icon in the status bar indicates the filter status of the repository panes. Possible values: ▪ OFF. Indicates that the object repositories are not filtered and all objects are shown. ▪ ON. Indicates a filter is active and that some objects may have been filtered out of the display.
Relevant tasks	"How to Compare Two Object Repositories" on page 1336
See also	"Understanding the Repository Panes" on page 1334

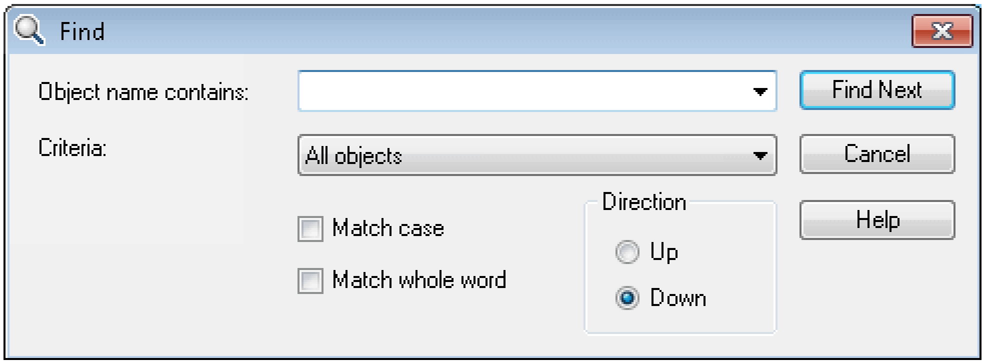
User interface elements are described below:

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

Find Dialog Box (Object Repository Comparison Tool)

Relevant for: GUI tests and components

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



To access	In the "Object Repository Comparison Tool Main Window" (described on page 1338), click the object repository pane that contains the required object and do one of the following: - Select **Navigate > Find**. - Click the **Find** 🔍 button in the toolbar.
Important information	After you set the Find options, you can close the Find dialog box and use toolbar buttons or shortcut keys to navigate to the next or previous node that matches your search criteria. For details, see "Menu Commands and Toolbar Buttons" on page 1339.
Relevant tasks	"How to Compare Two Object Repositories" on page 1336
See also	"Understanding the Repository Panes" on page 1334

User interface elements are described below:

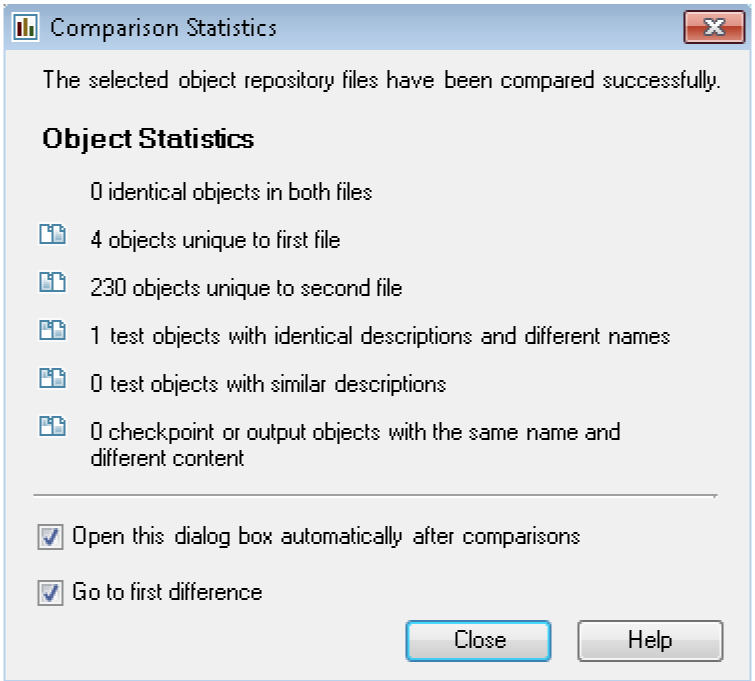
UI Elements	Description
Object name contains	The full or partial name of the object you want to find.

UI Elements	Description
Criteria	The criteria to use to refine your search. The following criteria are available: • **All objects** • **Unique objects** • **Partial match objects** • **Unique or partial match objects**
Match case	Instructs the Object Repository Comparison tool to distinguish between upper-case and lower-case characters in the search. When **Match case** is selected, the comparison finds only those occurrences with the exact capitalization match to the text you entered in the **Object name contains** box.
Match whole word	Instructs the Object Repository Comparison tool to search only for whole word occurrences that match the text you entered in the Find Objects dialog box, and not as part of larger words.
Direction	The direction from the current cursor location in which you want to search. The following options are available: • **Up** • **Down** The **Find** operation continues to search the entire file after it reaches the beginning or end of the object repository.
Find Next	Highlights the next object that matches the specified criteria in the object repository.

Comparison Statistics Dialog Box

Relevant for: GUI tests and components

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



The selected object repository files have been compared successfully.

Object Statistics

0 identical objects in both files

4 objects unique to first file

230 objects unique to second file

1 test objects with identical descriptions and different names

0 test objects with similar descriptions

0 checkpoint or output objects with the same name and different content

☑ Open this dialog box automatically after comparisons

☑ Go to first difference

To access	In the "Object Repository Comparison Tool Main Window" on page 1338, do one of the following: • Select **View** > **Statistics**. • Click the **Statistics** button in the toolbar.
Important information	The icons displayed for each comparison type in the object statistics are the same as those used in the object repository panes. For details, see "Understanding Object Differences" on the next page.
Relevant tasks	"How to Compare Two Object Repositories" on page 1336
See also	"Understanding the Repository Panes" on page 1334

User interface elements are described below:

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

Understanding Object Differences

The Comparison Tool automatically identifies objects during the comparison process by classifying them into one of the following types:

- **Identical.** Objects that are identical in both object repository files.

- **Unique to first file**, or **Unique to second file.** Objects that appear in only one of the object repository files.

- **Test objects with identical descriptions and different names.** Test objects that appear in both object repository files that have different names, but the same description properties and values.

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

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

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

- **Checkpoint or output objects with the same name and different content.** This option is relevant for checkpoint and output objects that are not completely identical for all settings.

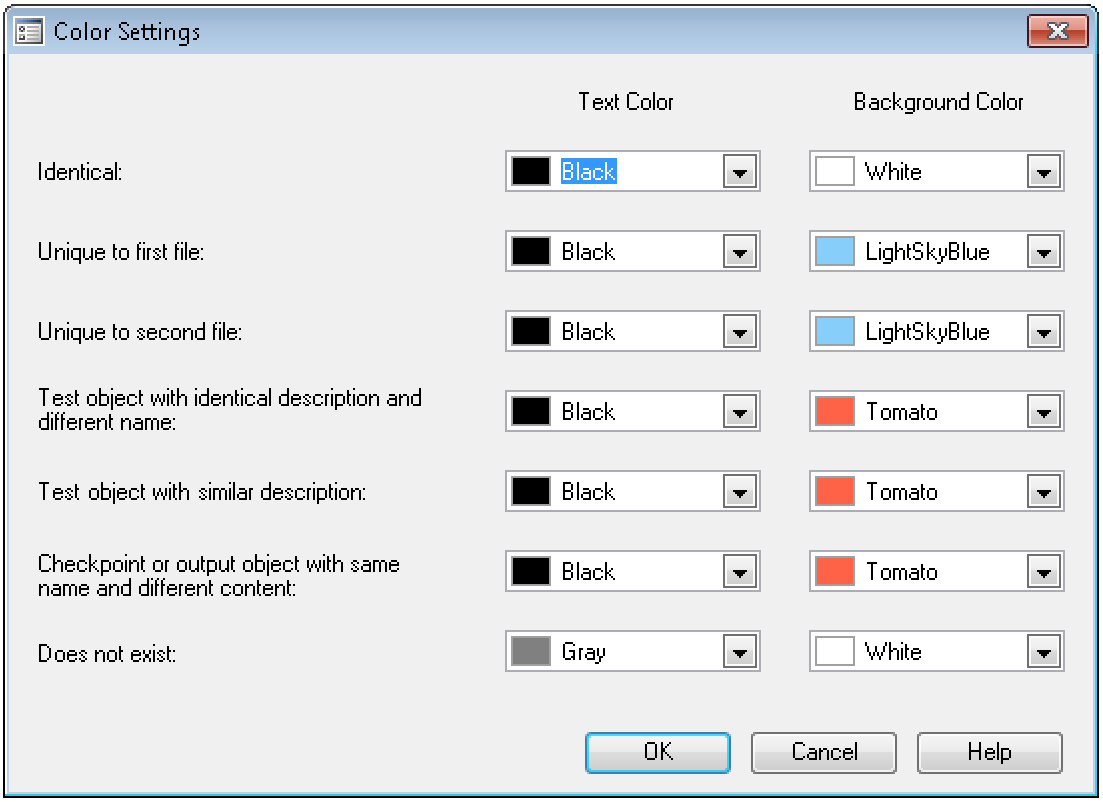
- **Does not exist.** Objects that do not exist in one of the repository files, but do exist in the other file.

Object differences can also be viewed in the "Object Repository Comparison Tool Main Window" on page 1338, according to settings defined in the "Color Settings Dialog Box (Object Repository Comparison Tool) " (described on page 1348).

Color Settings Dialog Box (Object Repository Comparison Tool)

Relevant for: GUI tests and components

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



	Text Color	Background Color
Identical:	Black	White
Unique to first file:	Black	LightSkyBlue
Unique to second file:	Black	LightSkyBlue
Test object with identical description and different name:	Black	Tomato
Test object with similar description:	Black	Tomato
Checkpoint or output object with same name and different content:	Black	Tomato
Does not exist:	Gray	White

To access	In the "Object Repository Comparison Tool Main Window" (described on page 1338), do one of the following: • Select **Tools > Color Settings**. • Click the **Color Settings** ▤ button in the toolbar.
Relevant tasks	"How to Compare Two Object Repositories" on page 1336
See also	"Understanding the Repository Panes" on page 1334

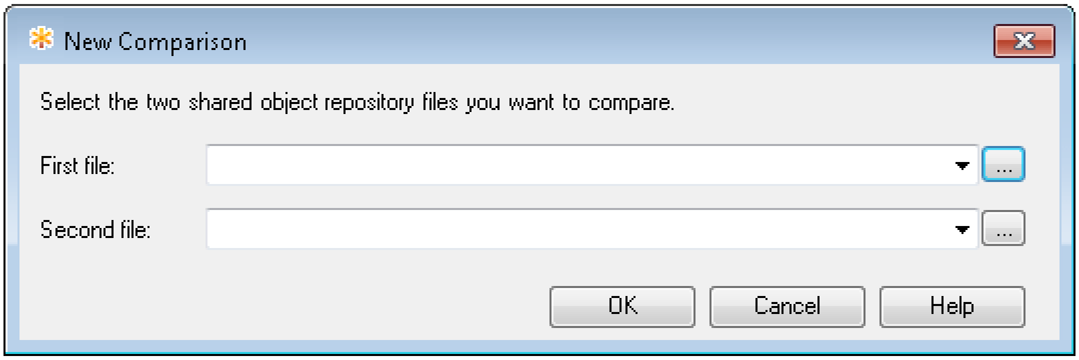
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<comparison types>	The list of comparison options on the left side of the dialog box. Difference types are described in "Understanding Object Differences" on page 1347.
Test Color	Displays the current color used for the object names matching the corresponding comparison type.
Background Color	Displays the current color used for the repository pane rows matching the corresponding comparison type.

New Comparison Dialog Box

Relevant for: GUI tests and components

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



New Comparison

Select the two shared object repository files you want to compare.

To access	In the "Object Repository Comparison Tool Main Window" (described on page 1338), do one of the following: • Select **File** > **New Comparison**. • Click the **New Comparison** button in the toolbar. **Note:**This dialog box also opens automatically when you initially open the "Object Repository Comparison Tool Main Window" on page 1338.

Important information	• The object repository files can be located in the file system or ALM. By default, the boxes display the last files selected for comparison. • If you want to compare an object repository that was last saved using a version of QuickTest earlier than version 9.0, you must first open and save the repository in the Object Repository Manager to update it to the new format. • If you want to change the color settings before comparing the object repositories, click **Cancel** to close the New Comparison dialog box, change the settings as described in "How to Compare Two Object Repositories" on page 1336, and then perform the comparison.
Relevant tasks	"How to Compare Two Object Repositories" on page 1336

User interface elements are described below:

UI Elements	Description
First file	The file path for the first object repository.
Second file	The file path for the second object repository.
❗	**Warning icon.** Displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. **Tip:** Position your pointer over the icon to see a tooltip explanation of the error. Supply an existing **.tsr** file with the correct path.

Chapter 48: Object Repository Merge Tool

Relevant for: GUI tests and components

This chapter includes:

Concepts

Object Repository Merge Tool Overview

Relevant for: GUI tests and components

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

This tool also enables you to merge objects from the local object repository of one or more actions or components into a shared object repository. For example, if UFT learned objects locally in a specific action in your test or in a component, you may want to add the objects to the shared object repository, so that they are available to all actions (even in different tests) or components that use that object repository.

When to Merge Shared Object Repositories

When you have multiple shared object repositories that contain test objects from the same area of your application, it may be useful to combine those test objects into a single object repository for easier maintenance. You could do this by manually moving or copying objects in the Object Repository Manager. However, if you have test objects in different object repositories that represent the same object in your application, and the descriptions for these objects in the different object repositories are not identical, it may be difficult to recognize and handle these conflicts.

The Object Repository Merge Tool helps you to solve the above problem by merging two selected object repositories for you, and providing options for addressing test objects with conflicting descriptions. Using this tool, you merge two shared object repositories (called the **primary** object repository and the **secondary** object repository), into a new, third object repository, called the **target** object repository. Objects in the primary and secondary object repositories are automatically compared, and then added to the target object repository according to configurable rules that define the defaults for how conflicts between objects are resolved.

After the merge process, the Object Repository Merge Tool provides a graphic presentation of the original objects in the primary and secondary object repositories, which remain unchanged, as well as the objects in the merged target object repository. Objects that had conflicts are highlighted. The conflict of each object that you select in the target object repository is described in detail. The Object Repository Merge Tool provides specific options that enable you to keep the default resolution for each conflict, or modify conflict resolutions individually, according to your requirements.

For more details, see "How to Merge Two Shared Object Repositories" on page 1356.

> **Note:** If you want to compare two shared object repositories without merging, you can use the Object Repository Comparison Tool. For details, see "Object Repository Comparison Tool Overview" on page 1334.

When to Update a Shared Object Repository from Local Object Repositories

You can update a shared object repository by merging local object repositories associated with specific components (via their application area), or with specific actions from one or more tests, into the shared object repository. The objects that are merged from the local object repositories are then available to any components that use that shared object repository, or to any actions that use that shared object repository in any tests.

In the merge process, the objects in the local object repository for the selected actions or components are moved to the target shared object repository (and then removed from the local object repository). The action or component steps then use the objects from the updated shared object repository.

You can view or change how conflicting objects are dealt with during the update process in the "Settings Dialog Box (Object Repository Merge Tool)" (described on page 1374).

If you choose to merge local object repositories from more than one action or component, UFT performs multiple merges, merging each action's or component's local object repository with the target object repository one at a time, for all the actions or components in the list. You can view and modify the results of each merge if necessary.

For more details, see "How to Update a Shared Object Repository From a Local Object Repository" on page 1358.

Object Conflicts

Relevant for: GUI tests and components

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

Conflicts between objects in the primary and secondary object repositories are resolved automatically by the Object Repository Merge Tool, according to the default resolution settings that you can configure before performing the merge. For details, see "Settings Dialog Box (Object Repository Merge Tool)" on page 1374.

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

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

Example

> An object in the primary object repository could have the same name as an object in the secondary object repository, but have a different description. You may have defined in the default settings that in this case, the object with the more generic object description, meaning the object with fewer properties, should be added to the target object repository. However, when you review the conflicts after the automatic merge, you could decide to handle a specific conflict differently, for example, by keeping both objects.

Changes that you make to the default conflict resolution can themselves affect the target object repository by causing new conflicts. In the above example, keeping both objects would cause a name conflict. Therefore, the target object repository is updated after each conflict resolution change and redisplayed.

The Object Repository Merge Tool identifies three possible conflict types:

Different Objects with the Same Name Conflict

Relevant for: GUI tests and components

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

You can resolve this conflict type by:

- Keeping the object added from the primary object repository only.
- Keeping the object added from the secondary object repository only.
- Keeping the objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Edit_1.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object from both files. The object that is added from the secondary file is renamed by adding an incremental numeric suffix to the name, for example, Edit_1. For details on changing the default settings, see "Settings Dialog Box (Object Repository Merge Tool)" on page 1374.

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

Identical Description Different Name Conflict (Test Objects Only)

Relevant for: GUI tests and components

A test object in the primary object repository and a test object in the secondary object repository have different names, but the same description properties and values.

You can resolve this conflict type by:

- Taking the test object name from the object in the primary object repository.
- Taking the test object name from the object in the secondary object repository.

- Ignoring the test object from the local object repository and keeping the test object from the shared object repository (when updating a shared object repository from a local object repository).

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object name from the primary source file. For details on changing the default settings, see "Settings Dialog Box (Object Repository Merge Tool)" on page 1374.

Similar Description Conflict (Test Objects Only)

Relevant for: GUI tests and components

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

You can resolve this conflict type by:

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

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the test object that has fewer identifying properties than the test object with which it conflicts. For details on changing the default settings, see "Settings Dialog Box (Object Repository Merge Tool)" on page 1374.

Tasks

How to Merge Two Shared Object Repositories

Relevant for: GUI tests and components

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

This task includes the following steps:

- "Prerequisites" below
- "Select the shared object repositories to merge " below
- "Analyze the initial merge results " on the next page
- "Analyze the detailed merge results " on the next page
- "Utilize additional tools to help you perform the comparison (Optional)" on the next page
- "Adjust object conflict resolutions" on the next page
- "Save the target object repository" on the next page
- "Associate the target object repository with actions or application areas" on page 1358

1. Prerequisites

- Determine the shared object repositories you want to compare. You generally merge two shared object repositories that contain objects from the same application, and that may have differences in objects or test object descriptions because the repositories were created at different times or under different circumstances.
- Make sure that a GUI test or component is in focus.
- Make sure that the Object Repository Manager window is open. For details on working with the Object Repository Manager, see "Object Repository Manager Main Window" on page 1297.
- Make sure the resolution and color settings are configured to match your needs. For details, see "Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 1375 and "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 1377.

2. Select the shared object repositories to merge

a. In the Object Repository Manager window, select **Tools > Object Repository Merge Tool** to open the Object Repository Merge Tool. The "New Merge Dialog Box" (described on page 1361) opens.

b. Specify the two object repository files you want to merge.

3. **Analyze the initial merge results**

   After the merge is complete, you can view the results summary in the "Merge Statistics Dialog Box (Object Repository Merge Tool)" (described on page 1370).

4. **Analyze the detailed merge results**

   Review and analyze the merge between the repositories in the "Object Repository Merge Tool Main Window" (described on page 1363).

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

   - Change the view presented by the Object Repository Merge Tool according to your working preferences, by dragging the edges of the panes to resize them, or selecting the appropriate option from the **View** menu, as described in "Object Repository Merge Tool Main Window" on page 1363.

   - Filter the objects and show only the objects that you want to view by using the "Filter Dialog Box (Object Repository Merge Tool)" (described on page 1367).

   - Locate one or more objects in a selected object repository whose name contains a specified string using the "Find Dialog Box (Object Repository Merge Tool)" (described on page 1368).

6. **Adjust object conflict resolutions**

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

   a. In the target object repository, select an object that had a conflict, as indicated by the icon to the left of the object name. The conflicting objects are highlighted in the source object repositories.

      A description of the conflict and the resolution method used by the Object Repository Merge Tool is described in the Resolution Options pane. A radio button for each possible alternative resolution method is displayed. For details on each of the conflict types, see "Object Conflicts" on page 1353.

   b. In the Resolution Options pane, select a radio button to choose an alternative resolution method. The target object repository is updated according to your selection and redisplayed.

   c. In the Resolution Options pane, click the **Previous Conflict** or **Next Conflict** buttons to jump directly to the next or previous conflict in the target object repository hierarchy.

7. **Save the target object repository**

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

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

8. **Associate the target object repository with actions or application areas**

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

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

Relevant for: GUI tests and components

This task describes how to move the objects from local object repositories to a shared object repository.

This task includes the following steps:

- "Prerequisites" below
- "Select the shared object repository and the local repositories that you want to merge into it" on the next page
- "Analyze the initial merge results" on the next page
- "Analyze the detailed merge results" on the next page
- "Utilize additional tools to help you perform the comparison (Optional)" on the next page
- "Adjust object conflict resolutions" on the next page
- "Save the target object repository" on page 1360

1. **Prerequisites**

   - Make sure that the shared object repository you want to update from the local object repositories is already associated with the repository actions or components.
   - Make sure the tests or components containing the local object repositories are not part of an open solution.
   - Make sure that a GUI test or component is in focus.
   - Make sure that the Object Repository Manager window is open. For details on working with

the Object Repository Manager, see "Object Repository Manager Main Window" on page 1297.

- Make sure the resolution and color settings are configured to match your needs. For details, see "Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 1375 and "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 1377.

2. Select the shared object repository and the local repositories that you want to merge into it

a. In the Object Repository Manager, open the shared object repository into which you want to merge the local repositories. If the object repository opened in read-only mode, select **File > Enable Editing**.

b. Select **Tools > Update from Local Repository** to open the "Update from Local Repository Dialog Box" (described on page 1381).

c. In the "Update from Local Repository Dialog Box", select the tests or components that contain the local object repositories you want to merge, and click **Update All**.

3. Analyze the initial merge results

View the initial merge results in the "Merge Statistics Dialog Box (Object Repository Merge Tool)" on page 1370.

4. Analyze the detailed merge results

Review and analyze the detailed merge results in the "Object Repository Merge Tool - Multiple Merge Window" on page 1378.

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

- Change the view presented by the Object Repository Merge Tool according to your working preferences, by dragging the edges of the panes to resize them, or selecting the appropriate option from the **View** menu, as described in "Object Repository Merge Tool - Multiple Merge Window" on page 1378.

- Filter the objects and show only the objects that you want to view by using the "Filter Dialog Box (Object Repository Merge Tool)" (described on page 1367).

- Locate one or more objects in a selected object repository whose name contains a specified string using the "Find Dialog Box (Object Repository Merge Tool)" (described on page 1368).

6. Adjust object conflict resolutions

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

a. In the target object repository, select an object that had a conflict, as indicated by the icon to the left of the object name. The conflicting object is highlighted in the local object repository.

A description of the conflict and the resolution method used by the Object Repository Merge Tool is described in the Resolution Options pane. A radio button for each possible alternative resolution method is displayed. For details on each of the conflict types, see "Object Conflicts" on page 1353.

b. In the Resolution Options pane, select a radio button to choose an alternative resolution method. The target object repository is updated according to your selection and redisplayed.

c. In the Resolution Options pane, click the **Previous Conflict** or **Next Conflict** buttons to jump directly to the next or previous conflict in the target object repository hierarchy.

7. Save the target object repository

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

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

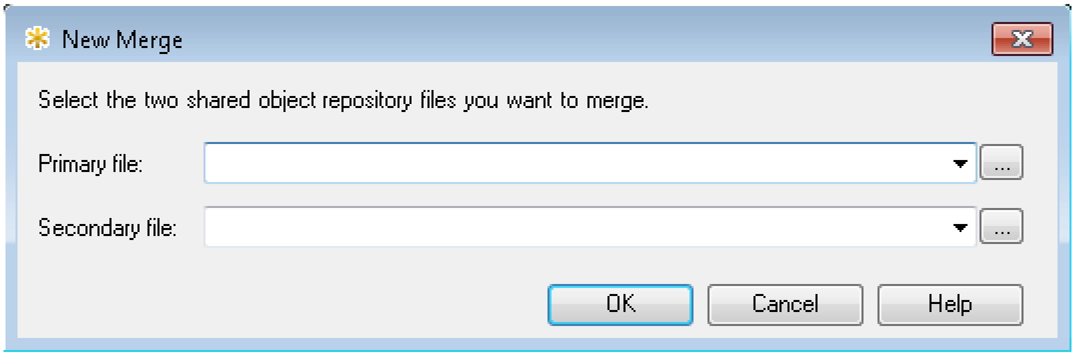
> **Note:** The objects that are merged into the shared object repository are **removed** from the local object repositories. The steps in the actions or components then use the objects from the updated shared object repository.

Reference

New Merge Dialog Box

Relevant for: GUI tests and components

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



Select the two shared object repository files you want to merge.

Primary file:

Secondary file:

To access	In the "Object Repository Merge Tool Main Window" (described on page 1363), do one of the following: - Select **File** > **New Merge**. - Click the **New Merge** button in the toolbar. **Note:** This dialog box also opens automatically when you initially open the "Object Repository Merge Tool Main Window" on page 1363.
Important information	- An object repository that is currently open by another user is locked. If you try to merge a locked file, a warning message displays, but you can still perform the merge because the merge process does not modify the source files. Note that changes made to the locked file by the other user may not be included in the merged object repository. - If this dialog box opens automatically, but you want to change the configured settings before merging the object repositories, click **Cancel** to close the New Merge dialog box, change the settings as described in "Settings Dialog Box (Object Repository Merge Tool)" on page 1374, and then open this dialog box again to perform the merge. - To improve efficiency of the merge process, select as your primary object repository the object repository in which you have invested the most effort, meaning the object repository with more objects, object properties, and values.
Relevant tasks	"How to Merge Two Shared Object Repositories" on page 1356

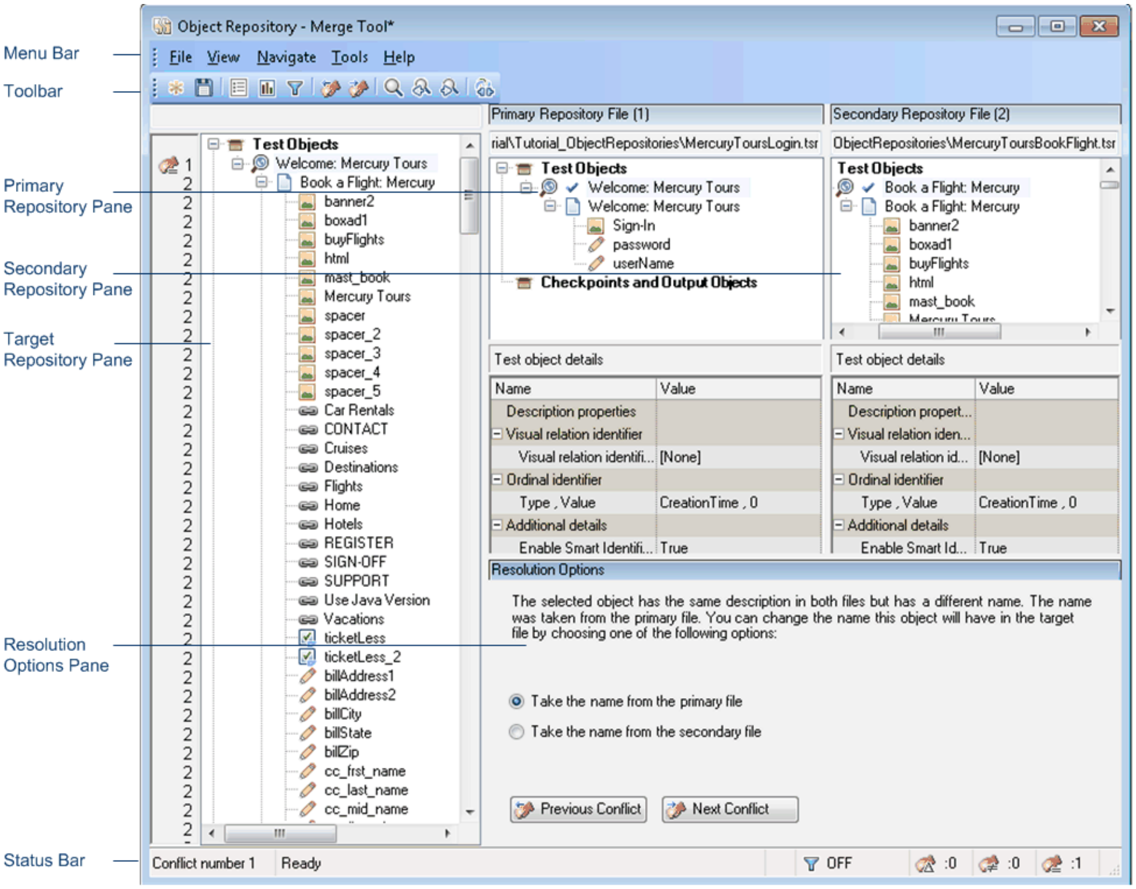
User interface elements are described below:

UI Elements	Description
Primary file	The file system or ALM path for the first object repository.
Secondary file	The file system or ALM path for the second object repository.
❗	**Warning icon.** Displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. **Tip:** Position your pointer over the icon to see a tooltip explanation of the error. Supply an existing **.tsr** file with the correct path.

Object Repository Merge Tool Main Window

Relevant for: GUI tests and components

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



Name	Value
Description properties	
Visual relation identifier	
Visual relation identifi...	[None]
Ordinal identifier	
Type , Value	CreationTime , 0
Additional details	
Enable Smart Identifi...	True

Name	Value
Description propert...	
Visual relation iden...	
Visual relation id...	[None]
Ordinal identifier	
Type , Value	CreationTime , 0
Additional details	
Enable Smart Id...	True

The selected object has the same description in both files but has a different name. The name was taken from the primary file. You can change the name this object will have in the target file by choosing one of the following options:

● Take the name from the primary file

○ Take the name from the secondary file

To access	In the **Object Repository Manager**, select **Tools > Object Repository Merge Tool**.
Important information	• You cannot work with the Object Repository Manager or the Object Repository Comparison Tool while the Object Repository Merge Tool is open. • Test objects that do not have description properties, such as Page or Browser objects, are compared by name only. If the same object is contained in both the source object repositories but with different names, they will be merged into the target object repository as two separate objects.
Relevant tasks	"How to Merge Two Shared Object Repositories" on page 1356

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

- "Menu Bar and Toolbar" below
- "Target Repository Pane" below
- "Primary and Secondary Repository Panes" on the next page
- "Resolution Options Pane" on the next page
- "Status Bar" on page 1366

Menu Bar and Toolbar

Displays menus with Object Repository Merge Tool commands. These commands are described in "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 1371.

Target Repository Pane

The target object repository pane displays a hierarchy of the objects, as well as their respective properties and values, that were merged from the primary and secondary object repositories. In the column to the left of the object hierarchy, the pane displays the source file of each object (**1** is displayed for the primary file and **2** for the secondary file), and an icon representing the type of conflict, if any.

When you save the target object repository, the file path is displayed above the object hierarchy.

> **Note:** To make it easier to see the status of an object at a glance, the text colors of the object names in the target object repository can be set according to their source and whether they caused a conflict. For details, see "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 1377.

Merging two object repositories can result in a target object repository containing a large number of objects. To make navigation and the location of specific objects easier in the target object repository pane, the Object Repository Merge Tool enables you to filter the objects in the pane and show only the objects that had conflicts that were resolved during the merge.

The target object repository pane provides the following functionality:

- When you select an object in the target object repository, the corresponding object in the primary and/or secondary source file hierarchy is located and indicated by a check mark.
- When you select an object in the target object repository, its properties and values are displayed in the **Object Properties - Target File** area at the bottom of the target object repository pane (**View > Target Repository Object Properties**).
- If the merge results in a conflict, an icon is displayed to the left of the conflicting object in the target object repository. You can see a tooltip description of the conflict type by positioning your pointer over the icon.
- When you right-click an object, a context-sensitive menu opens. You can expand an option or collapse the entire hierarchy in the target object repository, or, when applicable, you can change

the conflict resolution method and result.

- You can expand or collapse the hierarchy of the node by double-clicking a node. You can also expand or collapse the entire hierarchy in the target object repository by choosing **Collapse All** or **Expand All** from the **View** menu.

- You can jump directly to the next or previous conflict in the target object repository hierarchy by choosing **Next Conflict** or **Previous Conflict** from the **Navigate** menu, or by clicking the **Next Conflict** or **Previous Conflict** buttons in the toolbar or Resolution Options pane.

- You can locate one or more objects in the target object repository by using the "Find Dialog Box (Object Repository Merge Tool)" (described on page 1368).

- You can show or hide the target object repository object properties by choosing **View > Target Repository Object Properties**.

Primary and Secondary Repository Panes

The primary and secondary object repository panes display the hierarchies of the objects, and their properties and values, in the original source object repositories that you chose to merge. The file path is shown above each object hierarchy.

The panes provide the following functionality:

- You can expand or collapse the hierarchy of a selected item by double-clicking the item.

- You can view the properties and values of an object in the **Test object details** area by selecting it in the relevant pane.

- You can show or hide the panes by selecting or clearing **Primary Repository** or **Secondary Repository** in the **View** menu.

Resolution Options Pane

The Resolution Options pane provides information about any conflict encountered during the merge for the object selected in the target object repository. The pane also provides options that enable you to keep or change the conflict resolution method that was applied using the default resolution options.

The Resolution Options pane provides the following functionality:

- When you select a conflicting object in the target object repository, the pane displays a textual description of the conflict and the resolution method used by the Object Repository Merge Tool. A choice of alternative resolution methods is offered.

- You can select a radio button to choose an alternative resolution method for the conflict. Every time you make a change, the target object repository is automatically updated and is redisplayed.

- You can jump directly to the next or previous conflict in the target object repository hierarchy by clicking the **Previous Conflict** or **Next Conflict** buttons.

- For a local object repository merge, you can click the **Ignore Object** button to exclude a specific local object repository object from the merge process. The object remains in the local object repository when the merge is complete.

- You can show or hide the pane by selecting or clearing **Resolution Options** in the **View** menu.

Status Bar

The status bar shows the following UI elements (unlabeled elements are shown in angle brackets):

UI Elements	Description
Conflict number	The conflict number (if any) of the object selected in the target object repository pane.
<progress bar>	Displayed during the merge process. **Ready** is displayed when the merge is complete.
ON	**Filter**. Displays the filter status of the target object repository pane. Possible states: • **ON.** Indicated that a filter is currently in use. • **OFF.** indicates that the object repositories are not filtered and all objects are shown. **Note: Note:** Click the **Filter** icon to open the "Filter Dialog Box (Object Repository Merge Tool)" on the next page.
	ALM Connection. Displayed when UFT is connected to an ALM project.
⚠	**Similar Description Conflict**. For details on object conflicts, see "Object Conflicts" on page 1353.
≠	**Same Name Different Description Conflict**. For details on object conflicts, see "Object Conflicts" on page 1353.
=	**Same Description Different Name Conflict**. For details on object conflicts, see "Object Conflicts" on page 1353.

This section also includes:

- "Filter Dialog Box (Object Repository Merge Tool)" on the next page

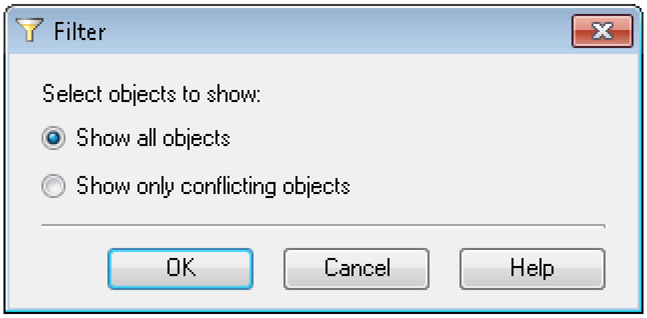
- "Find Dialog Box (Object Repository Merge Tool)" on page 1368

- "Merge Statistics Dialog Box (Object Repository Merge Tool)" on page 1370
- "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 1371

Filter Dialog Box (Object Repository Merge Tool)

Relevant for: GUI tests and components

This dialog box enables you to filter the target repository pane in the "Object Repository Merge Tool Main Window" (described on page 1363).



Filter

Select objects to show:

● Show all objects

○ Show only conflicting objects

To access	In the "Object Repository Merge Tool Main Window" on page 1363 , do one of the following: • Select **Tools** > **Filter**. • Click the **Filter** button in the toolbar.
Important information	The filter only affects which objects are displayed in the target object repository pane. It does not affect which objects are included in the target object repository.
Relevant tasks	"How to Merge Two Shared Object Repositories" on page 1356

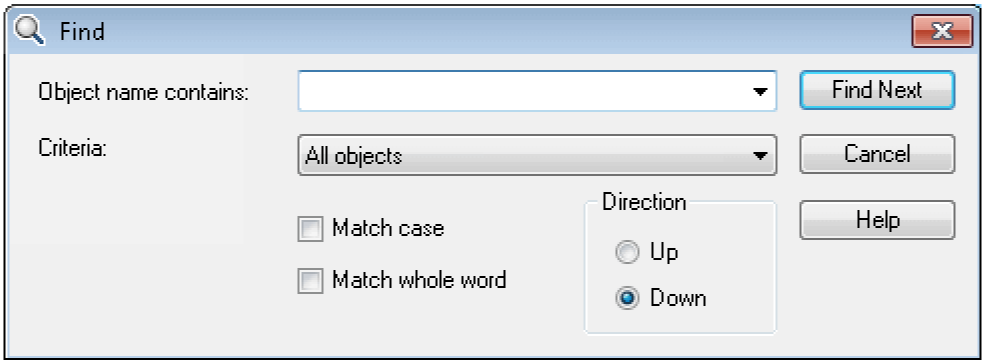
User interface elements are described below:

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

Find Dialog Box (Object Repository Merge Tool)

Relevant for: GUI actions and components

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



Object name contains:

Criteria: All objects

□ Match case

□ Match whole word

Direction

○ Up

● Down

To access	In the "Object Repository Merge Tool Main Window" (described on page 1363), do one of the following: - Select **Navigate > Find**. - Click the **Find** button in the toolbar.
Important information	After you set the Find options, you can close the Find dialog box and use menu commands or toolbar buttons to navigate to the next or previous node that matches your search criteria. For details, see "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 1371.
Relevant tasks	- "How to Merge Two Shared Object Repositories" on page 1356 - "How to Update a Shared Object Repository From a Local Object Repository" on page 1358
See also	"Object Conflicts" on page 1353

User interface elements are described below:

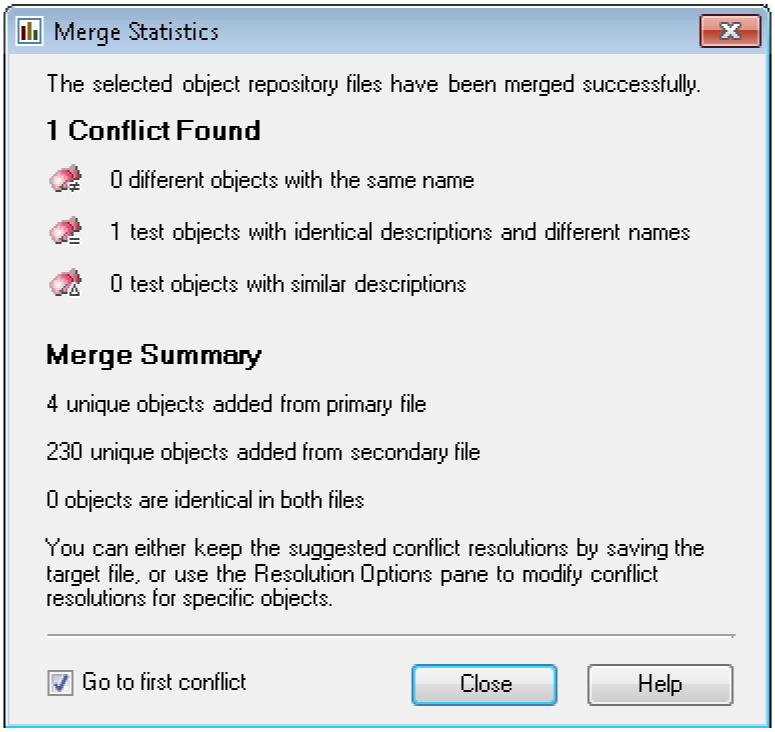
UI Elements	Description
Object name contains	The full or partial name of the object you want to find.

UI Elements	Description
Criteria	The criteria to use to refine your search. The following criteria are available: • **All objects** • **Objects from one source** • **Objects with conflicts** • **Objects with conflicts or from one source**
Match case	Instructs the Object Repository Merge Tool to distinguish between upper-case and lower-case characters in the search. When **Match case** is selected, UFT finds only those occurrences with the exact capitalization match to the text you entered in the **Object name contains** box.
Match whole word	Instructs the Object Repository Merge Tool to search only for whole word occurrences that match the text you entered in the Find Objects dialog box, and not as part of larger words.
Direction	The direction from the current cursor location in which you want to search. The following options are available: • **Up** • **Down** The **Find** operation continues to search the entire file after it reaches the beginning or end of the object repository.
Find Next	Highlights the next object that matches the specified criteria in the target object repository.

Merge Statistics Dialog Box (Object Repository Merge Tool)

Relevant for: GUI tests and components

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



The selected object repository files have been merged successfully.

1 Conflict Found

0 different objects with the same name

1 test objects with identical descriptions and different names

0 test objects with similar descriptions

Merge Summary

4 unique objects added from primary file

230 unique objects added from secondary file

0 objects are identical in both files

You can either keep the suggested conflict resolutions by saving the target file, or use the Resolution Options pane to modify conflict resolutions for specific objects.

☑ Go to first conflict

To access	In the "Object Repository Merge Tool Main Window" on page 1363 , do one of the following: • Select **View > Statistics**. • Click the **Statistics** button in the toolbar.
Important information	• The Statistics dialog box shown after performing an Update from Local Repository merge differs slightly from the dialog box shown above.
Relevant tasks	• "How to Merge Two Shared Object Repositories" on page 1356 • "How to Update a Shared Object Repository From a Local Object Repository" on page 1358

User interface elements are described below:

UI Elements	Description
Conflicts Found	The number and type of any conflicts between the objects added to the target object repository. Conflict types are described in "Object Conflicts" on page 1353.
Merge Summary	The number of objects merged from each object repository during the merge.
Go to first conflict	When selected, the Object Repository Merge Tool highlights the first conflict in the target object repository immediately after you close the Statistics dialog box.

Object Repository Merge Tool Menu Commands and Toolbar Buttons

Relevant for: GUI tests and components

The tables below describe commands available for:

- The "Object Repository Merge Tool Main Window" (described on page 1363), for merging two shared object repositories.
- The "Object Repository Merge Tool - Multiple Merge Window" (described on page 1378), for updating objects from local object repositories.

File Menu

	Command	Shortcut Key	Function
	New Merge (shared object repositories merge only)	CTRL+N	Enables you to specify two object repositories with which to perform a new merge operation.
	Save (shared object repositories merge only)	CTRL+S	Saves the merged shared object repository.
	Save As (shared object repositories merge only)		Opens the Save Shared Object Repository window, enabling you to specify a name, file type, and storage location for the merged shared object repository.

	Command	Shortcut Key	Function
	Save and Merge Next (update from local repository only)		When merging multiple local object repositories, saves the current merge and merges the next local object repository.
	Revert to Original Merged Files (update from local repository only)		Cancels any manual conflict resolution adjustments and returns the target object repository to the original state at the time of the merge.
	ALM Connection		Enables you to connect UFT to an ALM project. For details, see "ALM Connection Dialog Box" on page 737.
	Exit		Closes the Object Repository - Merge Tool window. If you did not yet save the merged repository, you are prompted to save it.

View Menu

	Command	Function
	Primary Repository	Displays the "Primary and Secondary Repository Panes" on page 1365 pane, containing a hierarchical view of the objects from the first source object repository that you chose to merge. Also displays the details for each object selected in this pane. For details, see "Primary and Secondary Repository Panes" on page 1365 and "New Merge Dialog Box" on page 1361.
	Secondary Repository (shared object repositories merge only)	Displays the "Primary and Secondary Repository Panes" on page 1365 pane, containing a hierarchical view of the objects from the second source object repository that you chose to merge. Also displays the details for each object selected in this pane. For details, see "Primary and Secondary Repository Panes" on page 1365 and "New Merge Dialog Box" on page 1361.
	Target Repository Object Properties	Displays the Object Properties - Target File pane, which displays the details for each test object selected in the target repository pane. For details, see "Target Repository Pane" on page 1364.
	Resolution Options	Displays the Resolution Options pane, which provides information about any conflict that occurred during the merge. For details, see"Resolution Options Pane" on page 1365 and"Object Conflicts" on page 1353.

	Command	Function
	Restore Default Layout	Restores the view that you saved using the **Set as Default Layout** option (described below). This is useful if you resize a pane, or show or hide specific panes and then want to restore your saved view. For details, see "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 1371.
	Set as Default Layout	Enables you to save the current view so that each time you open the Object Repository - Merge Too, this view is displayed. If you later modify this view by resizing panes, or showing or hiding them, you can restore your default view using the **Restore Default Layout** option (described above). For details, see "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 1371.
	Statistics	Opens the "Merge Statistics Dialog Box (Object Repository Merge Tool)" (described on page 1370), which describes how the files were merged, and the number and type of any conflicts that were resolved during the merge.
	Collapse All	Collapses the entire hierarchy in the Target Object Repository pane. **Tip:** You can collapse a single node by double-clicking it.
	Expand All	Expands the entire hierarchy in the Target Object Repository pane. **Tip:** You can expand a single node by double-clicking it.

Navigate Menu

	Command	Shortcut Key	Function
	Next Conflict	F4	Navigates to the next conflicting object in the merged object repository.
	Previous Conflict	SHIFT +F4	Navigates to the previous conflicting object in the merged object repository.
	Find	CTRL+F	Opens the "Find Dialog Box (Object Repository Merge Tool)" (described on page 1368).
	Find Next	F3	Navigates to the next object in the merged object repository according to the search specifications in the "Find Dialog Box (Object Repository Merge Tool)" (described on page 1368).
	Find Previous	SHIFT +F3	Navigates to the previous object in the merged object repository according to the search specifications in the Find dialog box.

Tools Menu

	Command	Function
	Settings	Opens the "Settings Dialog Box (Object Repository Merge Tool)" (described on page 1374), enabling you to: • Configure how the Object Repository Merge Tool deals with conflicting objects during a merge • Specify the text color of the object names displayed in the target object repository
	Filter	Opens the "Filter Dialog Box (Object Repository Merge Tool)" (described on page 1367), enabling you to show all of the objects in the Target Repository pane, or to show only the objects that had conflicts that were resolved during the merge.

Help Menu

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

Settings Dialog Box (Object Repository Merge Tool)

Relevant for: GUI tests and components

The Object Repository Merge Tool uses predefined settings when merging object repositories or when updating a shared object repository from local object repositories.

You can change these settings at any time. After you change the settings, all new merges are performed according to the new settings.

The Settings dialog box contains the following tabs:

- **Resolution Tab** (described on page 1375). Enables you to configure how the Object Repository Merge Tool deals with conflicting objects in the primary and secondary object repositories (or local and shared object repositories when updating a shared object repository from local object repositories).

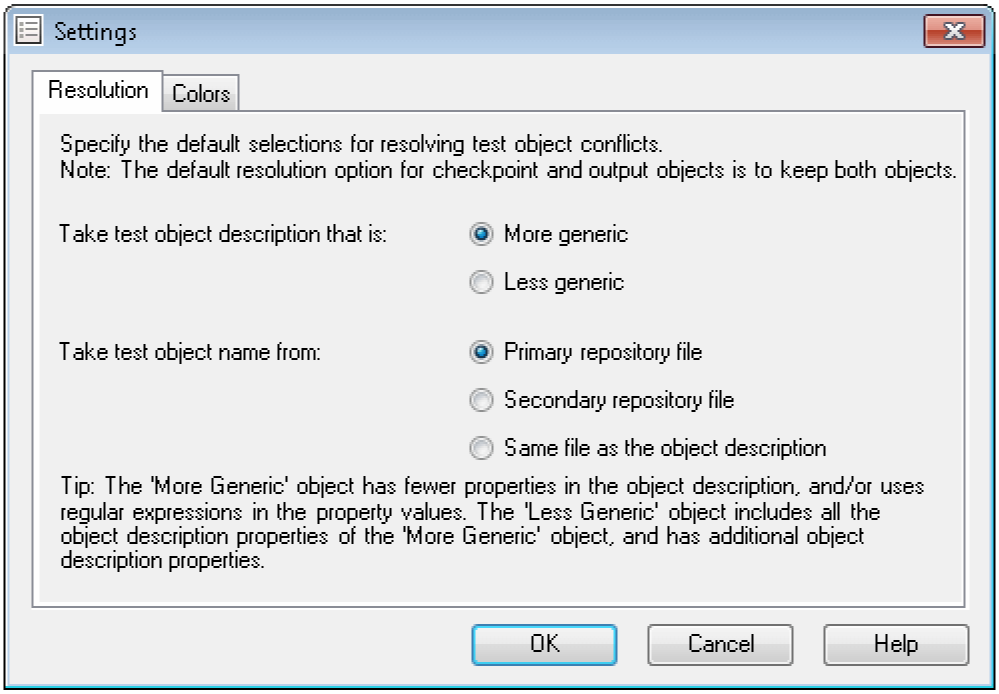
- **Colors Tab** (described on page 1377). Enables you to specify the text color of the object names that are displayed in the target object repository.

> **Tip:** If the New Merge dialog box opens when you open the Object Repository Merge Tool, and you want to change the settings before merging two object repositories, click **Cancel** to close the New Merge dialog box, change the settings as described in the next sections, and then open this dialog box again to perform the merge.

Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)

Relevant for: GUI tests and components

This tab enables you to configure how the Object Repository Merge Tool automatically deals with conflicting objects during the merge process or when performing an **Update from Local Repository** operation.



Specify the default selections for resolving test object conflicts.
Note: The default resolution option for checkpoint and output objects is to keep both objects.

Take test object description that is: ● More generic ○ Less generic

Take test object name from: ● Primary repository file ○ Secondary repository file ○ Same file as the object description

Tip: The 'More Generic' object has fewer properties in the object description, and/or uses regular expressions in the property values. The 'Less Generic' object includes all the object description properties of the 'More Generic' object, and has additional object description properties.

To access	In the "Object Repository Merge Tool Main Window" on page 1363 , do one of the following: • Select **Tools > Settings**, and select the **Resolution** tab. • Click the **Settings** button in the toolbar, and select the **Resolution** tab.

Important information	• The resolution settings are relevant only for test objects. Conflicts between checkpoint or output value objects with the same name but different content are always resolved by merging both objects into the new repository and renaming one of them. • When updating a shared object repository from a local object repository, the object repositories are referred to as the Local and Shared object repository. • If you make any change to the resolution settings while a merged object repository is open, you are asked whether you want to merge the open files again with the new settings. If you click **No**, the new settings will apply only to future merges.
Relevant tasks	• "How to Merge Two Shared Object Repositories" on page 1356 • "How to Update a Shared Object Repository From a Local Object Repository" on page 1358
See also	• "Object Conflicts" on page 1353 • "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on the next page

User interface elements are described below:

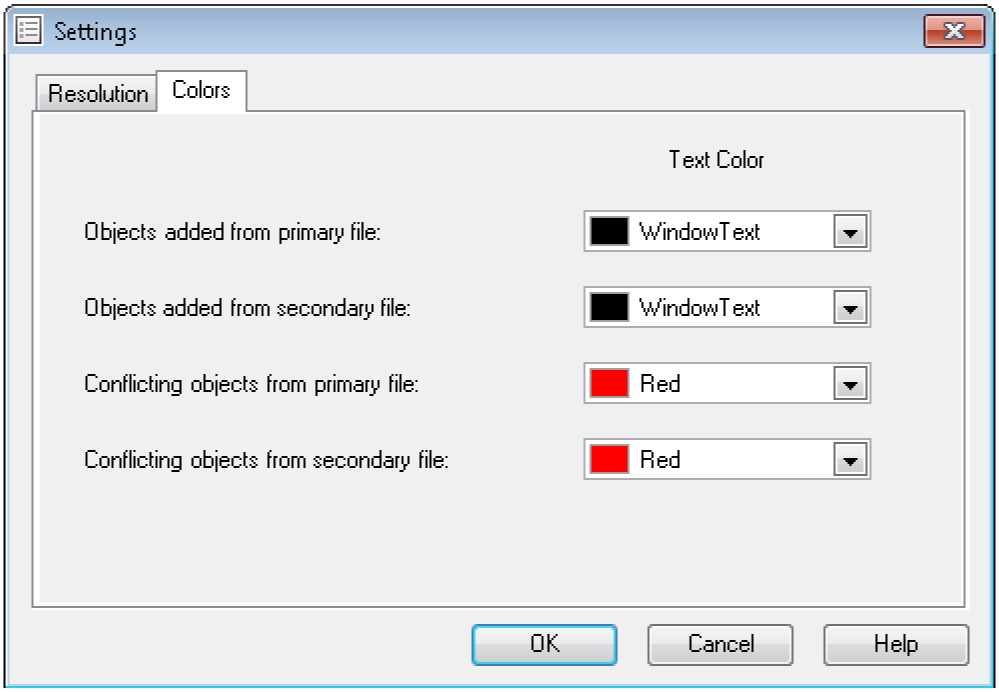
UI Elements	Description
Take test object description that is	Specifies how to resolve conflicts in which two test objects have the same name, but their descriptions differ. You can specify that the target object repository takes the test object description that is more generic or less generic, as follows: • **More generic**. Instructs the Object Repository Merge Tool to take the test object that has fewer identifying properties than the test object with which it conflicts, or uses regular expressions in its property values. This is the default setting. • **Less generic**. Instructs the Object Repository Merge Tool to take the test object that has all the identifying properties of the test object with which it conflicts, plus additional identifying properties.

UI Elements	Description
Take test object name from	Specifies how to resolve conflicts where two test objects have the same or similar descriptions, but their names differ. You can select the source from which the target test object repository takes the object name, as follows: • **Primary repository file**. The target object repository takes the test object name from the test object in the primary object repository. This is the default setting. (When updating a shared object repository from a local object repository, this option is for the **Local object repository**.) • **Secondary repository file**. The target object repository takes the test object name from the test object in the secondary object repository. (When updating a shared object repository from a local object repository, this option is for the **Shared object repository**.) • **Same file as the object description**. The target object repository takes the object name from the object in the same object repository from which it took the object description.

Colors Tab (Settings Dialog Box - Object Repository Merge Tool)

Relevant for: GUI tests and components

This tab enables you to specify the color in which object names are displayed in the target object repository according to their source, and whether they caused a conflict.



	Text Color
Objects added from primary file:	⬛ WindowText
Objects added from secondary file:	⬛ WindowText
Conflicting objects from primary file:	🟥 Red
Conflicting objects from secondary file:	🟥 Red

To access	In the "Object Repository Merge Tool Main Window" (described on page 1363), do one of the following: - Select **Tools > Settings**, and select the **Colors** tab. - Click the **Settings** button in the toolbar, and select the **Colors** tab.
Important information	When performing an **Update from Local Repository** operation, the options in the Colors tab of the"Settings Dialog Box (Object Repository Merge Tool)" (described on page 1374) also apply to objects added from the local (primary) and shared (secondary) object repositories.
Relevant tasks	"How to Merge Two Shared Object Repositories" on page 1356 "How to Update a Shared Object Repository From a Local Object Repository" on page 1358
See also	- "Object Conflicts" on page 1353 - "Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 1375

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

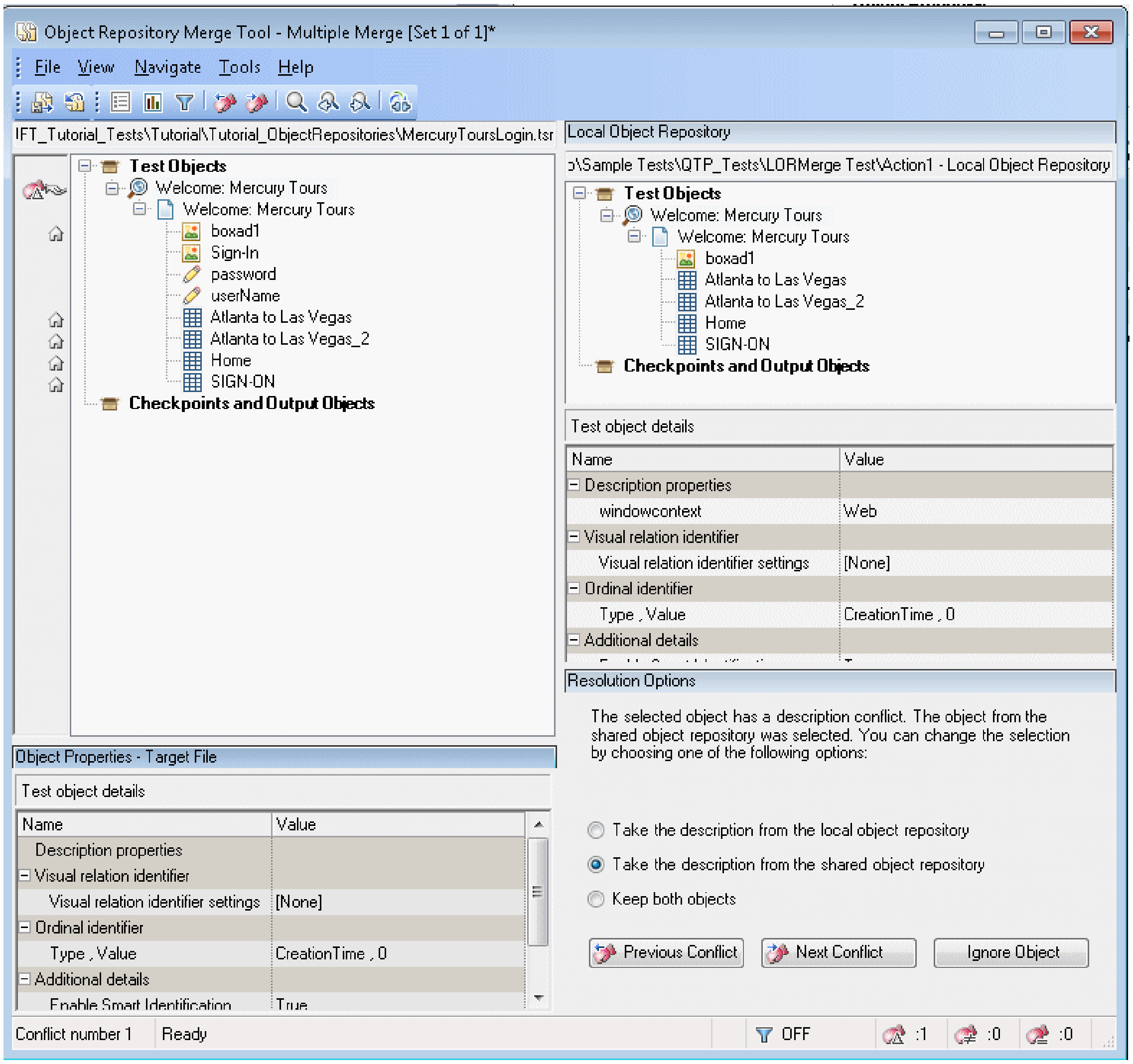
UI Elements	Description
<object criteria>	The list of object criteria on the left side of the dialog box.
Text Color	Displays the current color used for the object names that match the corresponding object criteria.

Object Repository Merge Tool - Multiple Merge Window

Relevant for: GUI tests and components

This window displays merge results of the selected local object repositories into the target shared object repository. The local object repositories are merged one by one into the shared object repository.

The active local object repository is treated as the primary object repository, and the shared object repository is treated as the target object repository.



Test object details

Name	Value
Description properties	
windowcontext	Web
Visual relation identifier	
Visual relation identifier settings	[None]
Ordinal identifier	
Type , Value	CreationTime , 0
Additional details	

Test object details

Name	Value
Description properties	
Visual relation identifier	
Visual relation identifier settings	[None]
Ordinal identifier	
Type , Value	CreationTime , 0
Additional details	
Enable Smart Identification	True

The selected object has a description conflict. The object from the shared object repository was selected. You can change the selection by choosing one of the following options:

○ Take the description from the local object repository

● Take the description from the shared object repository

○ Keep both objects

To access	In the "Object Repository Manager Main Window" (described on page 1297), select **Tools > Update from Local Repository**.

Important information	- If you specified more than one action or component in the "Update from Local Repository Dialog Box" (described on page 1381), UFT performs multiple merges, merging each local object repository with the target object repository one at a time. - The image above shows the merge results of the first merge (the first local object repository being merged into the shared object repository). - The number of each merge set in a multiple merge is displayed in the title bar, for example, [Set 2 of 3]. - When you click **Save and Merge Next**, the current merge is saved and cannot be modified without performing another merge. - The **Ignore Object** button is visible in the Merge Tool window only for a local object repository merge, and is only enabled when an object in the local object repository is selected. - If you are performing multiple merges, click the **Save and Merge Next** button in the Object Repository Merge Tool toolbar to perform the next merge (the next local object repository being merged into the shared object repository).
Relevant tasks	"How to Update a Shared Object Repository From a Local Object Repository" on page 1358

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

- "Menu Bar and Toolbar" below
- "Target Repository Pane" below
- "Primary Repository Pane" on the next page
- "Resolution Options Pane" on the next page
- "Status Bar " on the next page

Menu Bar and Toolbar

Displays menus of Object Repository Merge Tool commands. These commands are described in "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 1371.

Target Repository Pane

Displays the objects that were added from the local object repositories to the shared object repository.

At the left of each object in the target object hierarchy is an icon that indicates the source of the objects:

UI Elements	Description
⌂	Indicates that the object was added from the local object repository.
🤝	Indicates that the object already existed in the shared object repository.

Primary Repository Pane

Displays the objects in the local object repository that you are currently merging. For details, see "Update from Local Repository Dialog Box" below.

Resolution Options Pane

Provides source, conflict, and resolution details about the objects in the target object repository pane, and enables you to modify how a selected conflict is resolved. For details, see "Resolution Options Pane" on page 1365.

In the Resolution Options pane, you can select from one of the following options:

- Keep a specific object from the shared object repository and delete the conflicting object from the local object repository.
- Keep a specific object from the local object repository and delete the conflicting object from the shared object repository.
- Keep conflicting objects from both the shared object repository and the local object repository.
- **Ignore Object.** Exclude a specific local repository object from the merge process so that it is not included in the shared object repository. The object is removed from the shared object repository and grayed in the local object repository tree. It remains in the local object repository when the merge is complete.

> **Caution:** The **Ignore Object** operation cannot be reversed. To include the object again in the merge process, you must repeat the merge by clicking **Revert to Original Merged Files** in the toolbar.

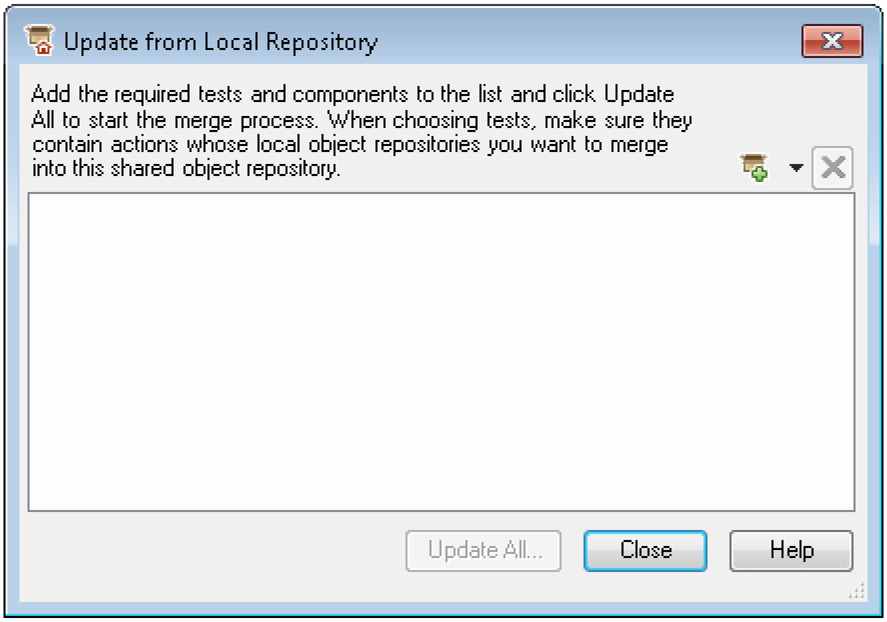
Status Bar

Provides source, conflict, and resolution details about the object selected in the target object repository pane, the filter status, and an icon legend. For details, see "Status Bar" on page 1366.

Update from Local Repository Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to select components and tests containing actions, whose local object repositories you want to merge into a shared object repository.



Add the required tests and components to the list and click Update All to start the merge process. When choosing tests, make sure they contain actions whose local object repositories you want to merge into this shared object repository.

To access	In the "Object Repository Manager Main Window" (described on page 1297), select **Tools** > **Update from Local Repository**.
Important information	• You can select multiple tests and components. • If you are currently connected to an ALM project, you can select tests and components from the file system or from ALM. • **For actions:** You can add only tests containing actions that are associated with the shared object repository you are updating and whose local object repositories contain objects. • **For components:** You can add components only if you are currently connected to the ALM project in which they are stored. The components must be associated (via their application area) with the shared object repository you are updating and their local object repositories must contain objects. • Before each merge, UFT checks whether the local object repository is in use by another user. If so, the local object repository is locked and the objects for the selected action or component cannot be moved to the target shared object repository. A warning message is displayed. The merge can be performed when the local object repository is no longer in use by the other user.
Relevant tasks	"How to Update a Shared Object Repository From a Local Object Repository" on page 1358

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

UI Elements	Description
<selected tests, actions, and components>	The list of actions (within the selected tests) and components that contain local object repositories to include in the merge.
	Add Tests or Components. Enables you to browse to a test or component and add it to the list. Click the down arrow ▾ to browse for tests or for components. (You can browse for components only when connected to ALM). **For actions:** If you select a test that contains multiple actions, only the actions that are associated with the shared object repository you are updating and whose local object repositories contain objects are added to the list.
Update All	Instructs the Object Repository Merge Tool to automatically merge the local object repositories with the shared object repository according to the preconfigured settings and opens the Object Repository Merge Tool. You can then modify any conflict resolutions if necessary. For details, see "Object Repository Merge Tool Main Window" on page 1363. **Note:** If the list contains multiple actions or components, merges are performed one after the other, and you can review the shared object repository and modify any conflict resolutions after each merge. For details, see "Object Repository Merge Tool - Multiple Merge Window" on page 1378.

Chapter 49: Virtual Objects

Relevant for: GUI tests and scripted GUI components

This chapter includes:

Concepts

Virtual Objects - Overview

Relevant for: GUI tests and scripted GUI components

Your application may contain objects that behave like standard objects but are not recognized by UFT. You can define these objects as virtual objects and map them to standard classes, such as a button or a check box. UFT emulates the user's action on the virtual object during the run session. In the run results, the virtual object is displayed as though it is a standard class object.

For example, suppose you want to test a Web page containing a bitmap that the user clicks. The bitmap contains several different hyperlink areas, and each area opens a different destination page. When you create the test or scripted component, the Web site matches the coordinates of the click on the bitmap and opens the destination page.

To enable UFT to click at the required coordinates during a run session, you can define a virtual object for an area of the bitmap, which includes those coordinates, and map it to the button class. When you run the test or scripted component, UFT clicks the bitmap in the area defined as a virtual object so that the Web site opens the correct destination page.

Virtual object collections are groups of virtual objects that are stored in the Virtual Object Manager under a descriptive name. For details, see "Virtual Object Manager Dialog Box" on page 1388.

The virtual object collections displayed in the Virtual Object Manager are stored on your computer and not with the tests or scripted components that contain virtual object steps. This means that if you use a virtual object in a step, the object is recognized during the run session only if it is run on a computer containing the appropriate virtual object definition. To copy your virtual object collection definitions to another computer, copy the contents of your **<UFT installation folder>\dat\VoTemplate** folder (or individual **.vot** collection files within this folder) to the same folder on the destination computer.

> **Note:** UFT does not support virtual objects for analog or low-level recording. For details on low-level recording, see "Creating Tests or Components" on page 2247.

How Virtual Objects are Defined and Recognized

Relevant for: GUI tests and scripted GUI components

UFT identifies a virtual object according to its boundaries. Marking an object's boundaries specifies its size and position on a Web page or application window. When you assign a test object as the parent of your virtual object, you specify that the coordinates of the virtual object boundaries are relative to that parent object. When you record a test or scripted component, UFT recognizes the virtual object within the parent object and adds it as a test object in the object repository so that UFT can identify the object during the run session. UFT also recognizes the virtual object as a test object when you add it manually to the object repository.

To perform an operation in the Active Screen on a marked virtual object, you must first record it, so that its properties are saved in the test object description in the object repository. If you perform an operation in the Active Screen on a virtual object that has not yet been recorded, UFT treats it as a standard object.

You can use virtual objects only when recording and running a test or scripted component. You cannot insert any type of checkpoint on a virtual object, or use the Object Spy to view its properties.

You can enable and disable recognition of virtual objects during recording, in the **General** pane of the GUI Testing tab in the Options dialog box (**Tools > Options > GUI Testing** tab **> General** node). For details, see "General Pane (Options Dialog Box > GUI Testing Tab)" on page 536.

During a run session, make sure that the application window is the same size and in the same location as it was during recording, otherwise the coordinates of the virtual object relative to its parent object may be different, and this may affect the success of the run session.

Tasks

How to Define Virtual Objects for Unsupported Objects in Your Test or Scripted Component

Relevant for: GUI tests and scripted GUI components

This task describes how to define virtual objects for objects that are not normally recognized by UFT.

This task includes the following steps:

- "Display the object containing the area you want to define as a virtual object" below
- "Use the Virtual Object wizard to define your virtual object" below

1. Display the object containing the area you want to define as a virtual object

With UFT open (but not in record mode), open your application and display the object containing the area you want to define as a virtual object.

You can define virtual objects only for objects on which UFT records **Click** or **DblClick** methods. Otherwise, the virtual object is ignored. For example, if you define a virtual object over the WinList object, the Select operation is recorded, and the virtual object is ignored. UFT does not support virtual objects for analog or low-level recording. For details on low-level recording, see "Frequently Asked Questions for GUI Testing" on page 2246.

2. Use the Virtual Object wizard to define your virtual object

Open the Virtual Object wizard (**Tools** > **Virtual Object** > **New Virtual Object**). The Virtual Object wizard includes the following pages:

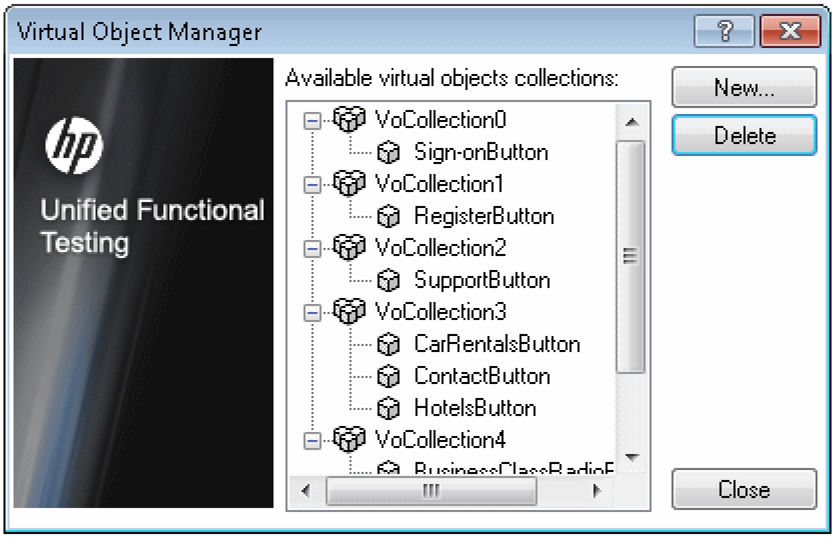
- "Map to a Standard Class Page (Virtual Object Wizard)", described on page 1390
- "Mark Virtual Object Page (Virtual Object Wizard)", described on page 1391
- "Object Configuration Page (Virtual Object Wizard)", described on page 1392
- "Save Virtual Object Page (Virtual Object Wizard)", described on page 1393

Reference

Virtual Object Manager Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to view and manage the virtual object collections defined on your computer. From the Virtual Object Manager, you can define and delete virtual objects and collections.



To access	1. 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 action node. 2. Select **Tools > Virtual Object > Virtual Object Manager.**
Important information	"How Virtual Objects are Defined and Recognized" on page 1385
Relevant tasks	"How to Define Virtual Objects for Unsupported Objects in Your Test or Scripted Component" on the previous page

User interface elements are described below:

UI Elements	Description
Available virtual object collections	Displays the virtual object collections defined on your computer and the virtual objects contained in each collection. Click the **+** and **-** signs next to a collection to view or hide the virtual objects defined in that collection.
New	Opens the Virtual Object Wizard, which guides you through the process of defining a new virtual object for a new or existing collection.
Delete	Deletes the selected virtual object or virtual object collection.

Virtual Object Wizard

Relevant for: GUI tests and scripted GUI components

This wizard enables you to define a virtual object by:

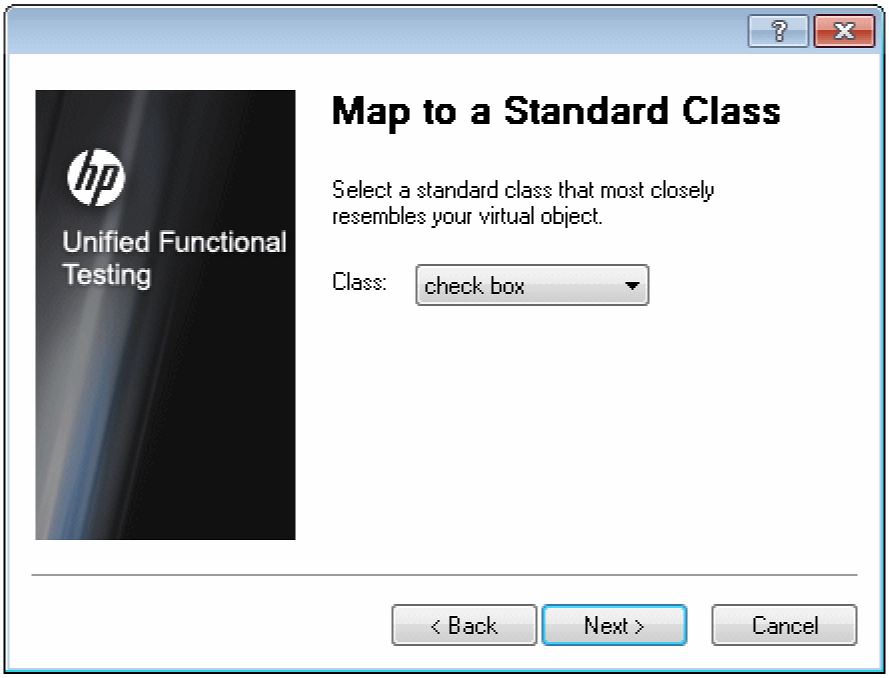
- Mapping it to a standard class
- Marking its boundaries
- Assigning it a parent object
- Specifying its name and grouping objects into collections

To access	1. 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 action node. 2. Use one of the following: ▪ Select **Tools > Virtual Object > New Virtual Object**. ▪ Select **Tools > Virtual Object > Virtual Object Manager.** From the Virtual Object Manager, click **New**.
Important information	"How Virtual Objects are Defined and Recognized" on page 1385
Relevant tasks	"How to Define Virtual Objects for Unsupported Objects in Your Test or Scripted Component" on page 1387
Wizard map	This wizard contains: **Welcome** > Map to a Standard Class (page 1390) > Mark Virtual Object (page 1391) > Object Configuration (page 1392) > Save Virtual Object (page 1393)

Map to a Standard Class Page (Virtual Object Wizard)

Relevant for: GUI tests and scripted GUI components

This wizard page enables you to configure a standard class for the virtual object.



Map to a Standard Class

Select a standard class that most closely resembles your virtual object.

Class: check box

Wizard map	"Virtual Object Wizard" contains: Welcome (page 1389) > **Map to a Standard Class** > Mark Virtual Object (page 1389) > Object Configuration (page 1392) > Save Virtual Object (page 1393)

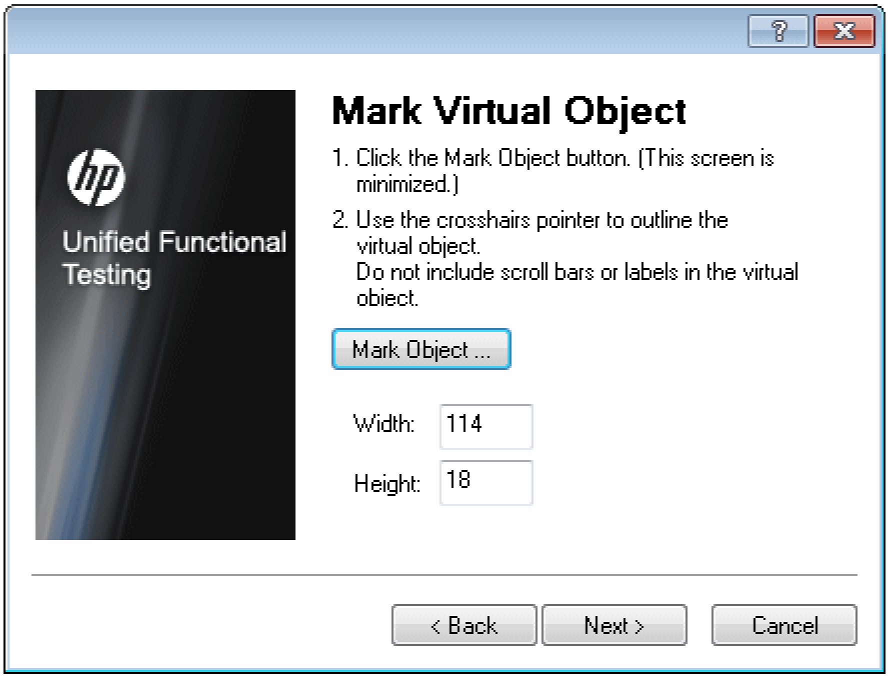
User interface elements are described below:

UI Elements	Description
Class	Specify a standard object class from the list. • For the list class, specify the number of rows in the virtual object. • For the table class, select the number of rows and columns.

Mark Virtual Object Page (Virtual Object Wizard)

Relevant for: GUI tests and scripted GUI components

This wizard page enables you to configure the size and location of the virtual object.



Mark Virtual Object

1. Click the Mark Object button. (This screen is minimized.)
2. Use the crosshairs pointer to outline the virtual object.
   Do not include scroll bars or labels in the virtual object.

Width: 114

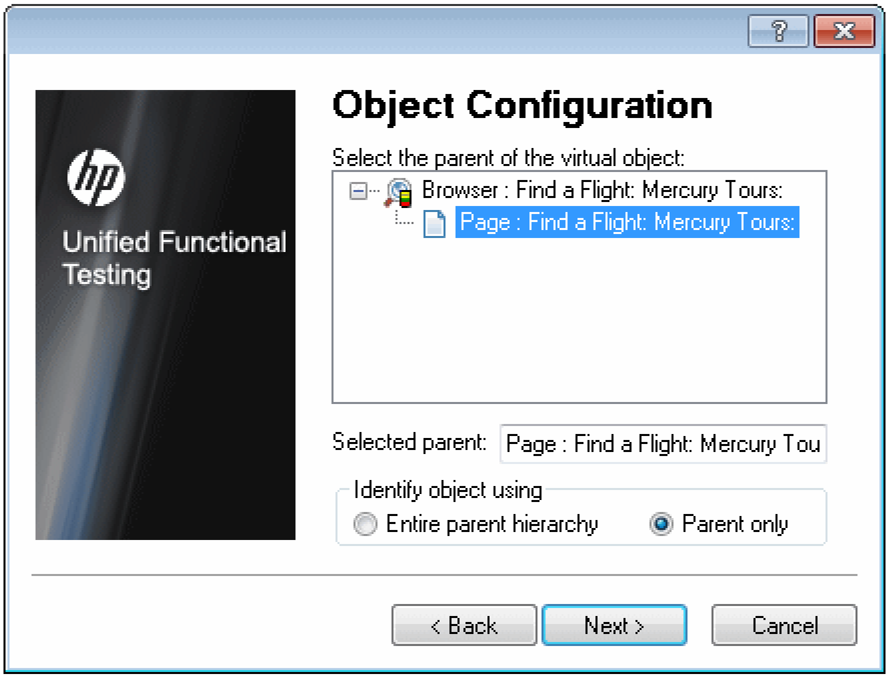
Height: 18

Important information	• Make sure that the virtual object does not overlap any other virtual object, as this may prevent UFT from identifying the virtual object during a run session. • To record and run tests or scripted components properly, you must ensure that the application window is the same size and in the same position as it was when you defined the virtual object.
Wizard map	"Virtual Object Wizard" contains: Welcome (page 1389) > Map to a Standard Class (page 1390) > **Mark Virtual Object** > Object Configuration (page 1392) > Save Virtual Object (page 1393)

Object Configuration Page (Virtual Object Wizard)

Relevant for: GUI tests and scripted GUI components

This wizard page enables you to configure an object as a parent of the virtual object.



Object Configuration

Select the parent of the virtual object:

- Browser : Find a Flight: Mercury Tours:
  - Page : Find a Flight: Mercury Tours:

Selected parent: Page : Find a Flight: Mercury Tou

Identify object using

○ Entire parent hierarchy ● Parent only

Wizard map	"Virtual Object Wizard" contains: Welcome (page 1389) > Map to a Standard Class (page 1390) > Mark Virtual Object (page 1391) > **Object Configuration** > Save Virtual Object (page 1393

User interface elements are described below:

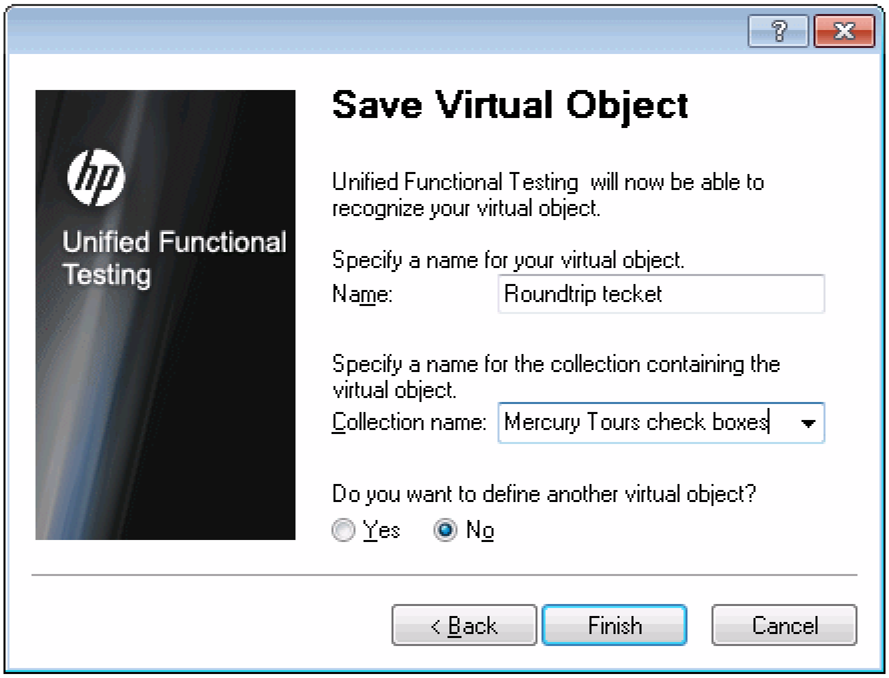
UI Elements	Description
Select the parent of the virtual object	Enables you to select an object in the tree as the parent object. The coordinates of the virtual object outline are relative to the parent object.
Selected parent	Indicates the name of the object selected as the parent object (read-only).

UI Elements	Description
Identify object using	- **Entire parent hierarchy.** Select this radio button to identify the virtual object in one occurrence only. UFT identifies the virtual object only if it has the exact parent hierarchy. For example, if the virtual object is defined using Browser("A").Page ("B").Image("C"), UFT does not recognize it if the hierarchy changes to Browser("X").Page("B").Image("C"). - **Parent only.** Select this radio button to identify all occurrences of the virtual object. UFT identifies the virtual object using its direct parent only, regardless of the entire parent hierarchy. For example, if the virtual object was defined using Browser("A").Page ("B").Image("C"), UFT recognizes the virtual object even if the hierarchy changes to Browser("X").Page("Y").Image("C").

Save Virtual Object Page (Virtual Object Wizard)

Relevant for: GUI tests and scripted GUI components

This wizard page enables you to configure a name and a collection for the virtual object. It also enables you to begin defining another virtual object.



Save Virtual Object

Unified Functional Testing will now be able to recognize your virtual object.

Specify a name for your virtual object.
Name: Roundtrip tecket

Specify a name for the collection containing the virtual object.
Collection name: Mercury Tours check boxes

Do you want to define another virtual object?
○ Yes ● No

Wizard map	"Virtual Object Wizard" contains: Welcome (page 13891389> Map to a Standard Class (page 1390 > Mark Virtual Object (page 1391) > Object Configuration (page 1392) > **Save Virtual Object**

User interface elements are described below:

UI Elements	Description
Name	The name of your new virtual object.
Collection name	The name of the collection in which your new virtual object is stored. Define virtual object collections in the "Virtual Object Manager Dialog Box" (described on page 1388).
Do you want to define another virtual object?	• Select **Yes** to enable the Next button and start the wizard again from the start. • Select **No** to enable the Finish button and close the wizard.

Chapter 50: Checkpoints in GUI Testing

Relevant for: GUI tests and components

This chapter includes:

Concepts

Checkpoints Overview

Relevant for: GUI tests and components

UFT enables you to add checks to your test or component. A **checkpoint** is a verification that compares the current value for specified properties or current state of other characteristics of an object with the expected value or characteristics. This helps you to identify whether your application is functioning correctly. For example, you can perform standard checkpoints to check that the actual object property values conform to the expected values, and you can perform bitmap checkpoints to check that the visible parts of your application are displayed correctly.

When you add a checkpoint, UFT inserts a checkpoint step to the current row in the Keyword View, and for tests and scripted components, also adds a **Check CheckPoint** statement in the Editor. By default, UFT names the checkpoint using the name of the test object on which the checkpoint was created. You can choose to specify a different name for the checkpoint or accept the default name.

When you run the test or component, UFT compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails. You can view the results of the checkpoint in the Run Results Viewer.

This section also includes:

Adding Existing Checkpoints to a Test

Relevant for: GUI tests and scripted GUI components only

UFT enables you to reuse existing checkpoints. When you create checkpoints, consider which checkpoints can be reused in multiple locations in your test or in multiple tests. For example:

- Checkpoints that check generic content or the state of your application may be useful in multiple locations.

- Checkpoints that check the content of a specific area of your application are generally useful in only one particular place in your test.

The following examples illustrate situations in which inserting an existing checkpoint may be useful:

- If each page of your application contains your organization's logo, you can reuse a bitmap checkpoint to verify each occurrence in the application.

- If your application contains multiple edit boxes, you can reuse a checkpoint to confirm the **enabled** status of these edit boxes throughout your test.

Simple and Advanced Mode

Relevant for: Keyword GUI components only

UFT provides two modes of checkpoints that you can insert using the relevant Checkpoint Properties dialog box—**Simple Mode** and **Advanced Mode**. Simple Mode enables you to check a basic set of properties and values. This type of checkpoint is visible and editable in both UFT and ALM. For details on the Checkpoint Properties dialog box, see "Checkpoint Properties Dialog Box" on page 1432.

In UFT, you can also view or edit advanced checkpoint properties that are not visible to users in ALM. You do this by inserting a checkpoint using Advanced Mode. If a checkpoint contains advanced properties, and an ALM user views its properties in ALM, a disclaimer opens indicating that some properties are checked but are not shown.

Checkpoint Types

Relevant for: GUI tests and components

You can insert the following checkpoint types to check objects in an application:

Checkpoint Type	Description
Standard Checkpoint	Checks property values of an object in your application. For example, you can check that a radio button is activated after it is selected or you can check the value of an edit box. Standard checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 2225). For details on standard checkpoints, see "Standard Checkpoints Overview" on page 1400.
Image Checkpoint (tests and scripted components only)	Checks the value of an image in your application. For example, you can check that a selected image's source file is correct. You create an image checkpoint by inserting a standard checkpoint on an image object. Image checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 2225). For details on image checkpoints, see "Standard Checkpoints Overview" on page 1400.

Checkpoint Type	Description
Accessibility Checkpoint (tests and scripted components only)	Identifies areas of your Web site that may not conform to the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines. For example, guideline 1.1 of the W3C Web Content Accessibility Guidelines requires you to provide a text equivalent for every non-text element. You can add an Alt property check to check whether objects that require the Alt property under this guideline, do in fact have this tag. Accessibility checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 2225). For details on accessibility checkpoints, see "Accessibility Checkpoints Overview" on page 1400.
Bitmap Checkpoint	Checks an area of your application as a bitmap. For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. Using the bitmap checkpoint, you can check that the map zooms in correctly. You can also check that a specific bitmap exists in your application. For example, you can check that your company logo is displayed anywhere on your Web page. You can create a bitmap checkpoint for any area in your application. Bitmap checkpoints are supported for all add-in environments. For details, see "Supported Checkpoints" on page 2225. For details on bitmap checkpoints, see "Bitmap Checkpoints Overview" on page 1401.
Database Checkpoint (tests and scripted components only)	Checks the contents of a database accessed by your application. For example, you can use a database checkpoint to check the contents of a database containing flight information for your Web site. Database checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 2225). For details on database checkpoints, see "Database Checkpoints Overview" on page 1405.
File Content Checkpoint (tests only)	Checks the text in a dynamically generated (or accessed) file. For example, suppose your application generates a .pdf. You can check that the correct text is displayed on specific lines in on specific pages in that .pdf. File content checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 2225). For details on file content checkpoints, see "File Content Checkpoints Overview" on page 1406.

Checkpoint Type	Description
Page Checkpoint (tests and scripted components only)	Checks the characteristics of a Web page. For example, you can check how long a Web page takes to load or whether a Web page contains broken links. You create a page checkpoint by inserting a standard checkpoint on a page object. Page checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 2225). For details on page checkpoints, see "Page Checkpoints Overview" on page 1407.
Table Checkpoint (tests and scripted components only)	Checks information within a table. For example, suppose your application contains a table listing all available flights from New York to San Francisco. You can add a table checkpoint to check that the time of the first flight in the table is correct. You create a table checkpoint by inserting a standard checkpoint on a table object. For details on table checkpoints, see "Table Checkpoints Overview" on page 1407. Table checkpoints are supported for all add-in environments that have a *Table test object. Table checkpoints are also supported for some list view objects, such as WinListView and VbListView, as well as other list view objects in add-in environments. For details, see "Supported Checkpoints" on page 2225.
Text Checkpoint (tests and scripted components only)	Checks that a text string is displayed in the appropriate place in an application. For example, suppose a Web page displays the sentence Flight departing from New York to San Francisco. You can create a text checkpoint that checks that the words "New York" are displayed between "Flight departing from" and "to San Francisco". Text checkpoints are supported for most add-in environments (see "Supported Checkpoints" on page 2225). For details on text checkpoints, see "Checking Text Overview " on page 1408.

Checkpoint Type	Description
Text Area Checkpoint (tests and scripted components only)	Checks that a text string is displayed within a defined area in a Windows-based application, according to specified criteria. For example, suppose your Visual Basic application has a button that says View Doc <Num>, where <Num> is replaced by the four digit code entered in a form elsewhere in the application. You can create a text area checkpoint to confirm that the number displayed on the button is the same as the number entered in the form. Text area checkpoints are supported for all Windows-based environments, such as Standard Windows, Visual Basic, and ActiveX add-in environments (see "Supported Checkpoints" on page 2225). Text area checkpoints are also supported for some other add-in environments, such as Java. For details on text area checkpoints, see "Checking Text Overview " on page 1408.
XML Checkpoint (tests and scripted components only)	Checks the data content of .xml documents in ,xml files or .xml documents in Web pages and frames. For details on XML checkpoints, see "XML Checkpoints Overview" on page 1416 The **XML Checkpoint (Web Page/Frame)** option is supported for the Web add-in environment. The **XML Checkpoint** option is supported for all add-in environments (see "Supported Checkpoints" on page 2225).

Standard Checkpoints Overview

Relevant for: GUI tests and components

You can check the object property values in your application using standard checkpoints. Standard checkpoints compare the expected values of object properties to the object's current values during a run session. You can create standard checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

You can check that a specified object in your application has the property values you expect, by adding a standard checkpoint step to your test or component while recording or editing it. To set the options for a standard checkpoint, you use the "Checkpoint Properties Dialog Box" (described on page 1432).

For tests and scripted components: You can also use standard checkpoints to perform checks on images, tables, Web page properties, and other objects within your application.

Accessibility Checkpoints Overview

Relevant for: GUI tests and scripted GUI components

You can add accessibility checkpoints to help you quickly identify areas of your Web site that may not conform to the W3C Web Content Accessibility Guidelines.

The Section 508 criteria for Web-based technology and information systems are based on access guidelines developed by the Web Accessibility Initiative of the World Wide Web Consortium (W3C).

You can add automatic accessibility checkpoints to each page in your test, or you can add individual accessibility checkpoints to individual pages or frames.

Accessibility checkpoints are not supported for keyword components.

Automatic Accessibility Checkpoints

You can instruct UFT to create automatic accessibility checkpoints for every page in all tests. For details, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

Individual Accessibility Checkpoints

If you do not select to add accessibility checkpoints automatically while recording, you can add an accessibility checkpoint while recording or editing your test.

For details, see:

- "How to Insert a Checkpoint in a GUI Test or Component" on page 1419
- "Checkpoint Properties Dialog Box" on page 1432

Bitmap Checkpoints Overview

Relevant for: GUI tests and components

UFT enables you to check that the visible parts of your application are displayed correctly by comparing bitmaps of objects in your application to bitmaps captured previously and stored with the test or component.

You can create bitmap checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

Bitmap checkpoints enable you to do the following:

- **Compare an entire object or areas within an object.** For example, suppose you have a Web site that can display a map of a city that the user specifies. The map has control keys for zooming. You can zoom in on a map, and then insert a bitmap checkpoint on the zoomed-in map to check that the map zooms in correctly.
- **Locate a specified image within an object.** For example, suppose you want to check that your company logo is displayed on your Web page. You can either select the logo in the actual Web page, or load a bitmap file containing the logo from your computer.

For details on defining checkpoint settings, see "Checkpoint Properties Dialog Box" on page 1432.

How UFT Compares Bitmaps

When you create a bitmap checkpoint, UFT captures the **visible** part of the specified object as a bitmap and inserts a checkpoint in the test or component. (UFT does not capture any part that is scrolled off the screen, or hidden by another object, for example.)

You can specify areas of the object to ignore or include in the checkpoint. For example, if your Web page includes a dynamic counter that may cause the checkpoint to fail, you can instruct UFT to ignore it during the run session by excluding the area in which it is located from the comparison.

When you run the test or component, UFT captures a bitmap of the actual object in the application and compares this runtime bitmap (or the selected areas within it) with the bitmap stored in the checkpoint. You can fine-tune this comparison by defining tolerance settings in the checkpoint. For details, see "Fine-Tuning the Bitmap Comparison" on the next page.

If there are differences, UFT saves the runtime bitmap and displays it next to the expected bitmap in the details pane of the Run Results Viewer. In the Run Results Viewer you can also view a bitmap that reflects the difference between the two bitmaps, to assist you in identifying the nature of the discrepancy.

How UFT Locates Bitmaps

When you create a bitmap checkpoint, you select a bitmap that you want UFT to locate during a run session. You can either select an area within the bitmap captured from your application, or load a bitmap file from the file system and store it in the checkpoint.UFT then inserts the checkpoint in the test or component.

When you run the test or component, UFT searches for the specified bitmap within the runtime bitmap captured from the application. You can fine-tune this search by defining similarity criteria in the checkpoint. For details, see "Fine-Tuning the Bitmap Comparison" on the next page.

If the specified bitmap is not found, the checkpoint fails, and UFT displays the specified bitmap next to the actual (runtime) bitmap in the Details pane of the Run Results Viewer.

The Run Results Viewer also displays the coordinates of the nearest candidate bitmap within the actual bitmap, along with the similarity percentage used to find the candidate.

When to Use the Option to Locate a Bitmap Within the Runtime Bitmap

Suppose your application includes a specific bitmap that you want to make sure exists when you run your steps. The bitmap you want to check may not always be located in the same place in your application.

If you attempt to compare the expected bitmap with the runtime bitmap, and the location of the expected bitmap changes, then the checkpoint may fail. Therefore, you can define the specific bitmap that you want to locate, and UFT searches for that bitmap anywhere in the object on which you define the checkpoint, relative to the top-left corner of the object.

How UFT Displays Bitmap Checkpoint Results

By default, UFT only displays expected, actual, and difference bitmaps in the Run Results Viewer for checkpoints that fail. If a bitmap checkpoint is configured such that the checkpoint can pass even if the expected and actual bitmaps are not identical, you can configure the run results to display the captured actual, expected and difference bitmaps for bitmap checkpoints that pass, too. For details, see "Screen Capture Pane (Options Dialog Box > GUI Testing Tab)" on page 557.

The results of bitmap checkpoints may be affected by factors such as operating system, screen resolution, and color settings.

For details on run results of a checkpoint, see the section on Bitmap Checkpoint Results in the *HP Run Results Viewer User Guide*.

Fine-Tuning the Bitmap Comparison

Relevant for: GUI tests and components

When running a bitmap checkpoint, UFT compares the area that you are checking in the application with the bitmap stored in the checkpoint, pixel by pixel. By default, if any pixels are different, the checkpoint fails. The advanced settings in the Bitmap Checkpoint Properties dialog box (described on page 1432) provides various options for fine-tuning the bitmap comparison:

- When comparing bitmaps, you can adjust the comparison to enable the checkpoint to pass even if the bitmaps are not identical by setting the **RGB tolerance** and **Pixel tolerance** options described below.

- When locating a specified bitmap within the runtime bitmap, you can use **similarity** to enable the checkpoint to pass, even if the exact bitmap is not found in your application. For details on similarity, see "Image Similarity".

- You can also use a **custom comparers** to carry out a bitmap checkpoint. A custom comparer is a COM object that you or a third party can develop to run the bitmap comparison in the checkpoint, according to a more specific algorithm. For details on using the **Comparer** option, see "Custom Comparers".

RGB Tolerance

Note: This functionality is available only when comparing expected bitmaps with runtime bitmaps. It is not available when locating a specified bitmap within the runtime bitmap.

The RGB (Red, Green, Blue) tolerance determines the percent by which the RGB values of the pixels in the runtime bitmap can differ from those of the expected bitmap and allow the checkpoint to pass. (The RGB tolerance option is limited to bitmaps with a color depth of 24 bits.)

For example, a bitmap checkpoint on identical bitmaps could fail if different display drivers are used when you create your checkpoint and when you run your test. Suppose one display driver displays the color white as RGB (255, 255, 255) and another driver displays the color white as RGB (231, 231, 231). The difference between these two values is about 9.4%. By setting the **RGB tolerance** to 10%, your checkpoint will pass when running your test with either of these drivers.

Note: UFT applies the RGB tolerance settings when comparing each pixel in the expected and runtime bitmaps. The Red, Green, and Blue values for each pixel are compared separately. If any of the values differs more than the tolerance allows, the pixel fails the comparison.

Pixel Tolerance

Note: This functionality is available only when comparing expected bitmaps with runtime bitmaps. It is not available when locating a specified bitmap within the runtime bitmap.

The pixel tolerance determines the number or percentage of pixels in the runtime bitmap that can differ from those in the expected bitmap and allow the checkpoint to pass.

For example, suppose the expected bitmap has 4000 pixels. If you define the pixel tolerance to be 50 and select the **Pixels** radio button, up to 50 pixels in the runtime bitmap can be different from those in the expected bitmap and the checkpoint passes. If you define the pixel tolerance to be 5 and select the **Percent** radio button, up to 200 pixels (5 percent of 4000) in the runtime bitmap can be different from those in the expected bitmap and the checkpoint passes.

Using Both RGB and Pixel Tolerances

> **Note:** This functionality is available only when comparing expected bitmaps with runtime bitmaps. It is not available when locating a specified bitmap within the runtime bitmap.

If you define both RGB and pixel tolerances, the RGB tolerance is calculated first. The pixel tolerance then defines the maximum number of pixels that can fail the RGB criteria and allow the checkpoint to pass.

For example, suppose you define an RGB tolerance of 10 percent and a pixel tolerance of 5 percent, for a bitmap that has 4000 pixels.

For the checkpoint to pass, each pixel in the runtime bitmap must have RGB values that are no greater than or no less than 10 percent of the RGB values of the expected bitmap. If that criterion fails, UFT checks that the number of pixels that failed are less than 200. If that criterion passes, the checkpoint passes.

Image Similarity

> **Note:** This functionality is available only when locating a specified bitmap within the runtime bitmap. It is not available when comparing expected bitmaps with runtime bitmaps.

Image similarity settings can enable a checkpoint to pass, even if the exact bitmap is not found in your application. UFT attempts to locate the specified bitmap in the runtime bitmap of the object in your application during the run session. If UFT locates an exact match to the specified bitmap, then the checkpoint passes.

If an exact match cannot be found and you specified less than 100% in the **Similarity** option in the advanced settings of the "Checkpoint Properties Dialog Box" (described on page 1432), UFT adjusts the comparison according to the similarity level. If the possible candidate has a similarity that is equal to or greater than the percentage that you defined, the checkpoint passes.

Custom Comparers

A custom comparer is a COM object that you or a third party can develop to run the bitmap comparison in the checkpoint according to a more specific algorithm. If you use a custom comparer to perform the bitmap checkpoint, UFT sends the comparer two bitmaps to compare: A screen capture of the object, created with the checkpoint and saved as the expected bitmap, and a screen capture of the object as it appears in the application during the run session. The comparer then compares these two bitmaps according to the specifications in its algorithm. If you use a custom comparer, you cannot use the "Checkpoint Properties Dialog Box" (described on page 1432) to specify tolerance or similarity settings, or areas of the object to compare or ignore.

If one or more custom comparers are installed and registered on the UFT computer, the "Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)" (described on page 1463) includes a **Comparer** option.

The **Comparer** option enables you to select the UFT default comparer or a custom comparer that performs the bitmap comparison according to your testing requirements. For an example of when it can be useful to create a custom comparer, see "Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario" on page 2199. For details on developing or installing custom comparers, see "Developing Custom Comparers for Bitmap Checkpoints (GUI Testing)" on page 2198.

If you select a custom comparer, some of the options in the Bitmap Checkpoint Properties Advanced Settings dialog box are different. For details, see "Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)" on page 1463.

Database Checkpoints Overview

Relevant for: GUI tests and scripted GUI components

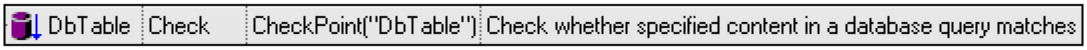
You can use database checkpoints to check databases accessed by your application, and to detect defects. To do this, you define a query on your database. Then you create a database checkpoint that checks the results of the query.

You can define a database query in the following ways:

- Using Microsoft Query. You can install Microsoft Query from the custom installation of Microsoft Office.
- By manually defining an SQL statement.

You create a database checkpoint based on the results of the query (**result set**) you defined on a database. You can create a check on a database to check the contents of the entire result set, or a part of it. UFT captures the current data from the database, saves this information as **expected data**, and inserts a database checkpoint step.

This checkpoint is displayed in the Editor as a **DbTable.Check CheckPoint** statement and as a step in the Keyword View.



DbTable	Check	CheckPoint("DbTable")	Check whether specified content in a database query matches

When you create a new database checkpoint, all cells contain a blue check mark, indicating they are selected for verification. You can select to check the entire results set, specific rows, specific columns, or specific cells. UFT checks only cells containing a check mark.

You can also specify the way UFT identifies the selected cells. For example, suppose you want to check the data that is displayed in the first row and second column in the "Checkpoint Properties Dialog Box" (described on page 1432). However, you know that each time you run your test or scripted component, it is possible that the rows may be in a different order, depending on the sorting that was performed in a previous step. Therefore, rather than finding the data based on row and column numbers, you may want UFT to identify the cell based on the column name and the row containing a known value in a **key column**.

During the run session, the database checkpoint compares the current data in the database to the expected data defined in the "Checkpoint Properties Dialog Box" on page 1432 If the expected data and the current results do not match, the database checkpoint fails.

> **Note:**
>
> - You can modify the expected data of a database checkpoint before a run session. You can also make changes to the query in an existing database checkpoint. This can be useful if you move the database to a new location on the network.
> - You can view the results of the checkpoint in the Run Results Viewer. For details, see the section on database checkpoint results in the *HP Run Results Viewer User Guide*.
> - For details on creating database checkpoints, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

File Content Checkpoints Overview

Relevant for: GUI tests and scripted GUI components

You can use file content checkpoints to compare the textual content of a file that is generated during a run session with the textual content of a source file. This enables you to verify that the generated file contains the expected results. For example, you may want to verified that a PDF file generated during a run session displays the local corporate address at the top of every page.

You can perform a checkpoint on text in one line, multiple lines, or the entire document, as needed. You can also specify what to ignore. For example, if you expect certain lines or areas in the file to change, you can exclude them from the checkpoint.

When you select a source document to compare, UFT converts a copy of this document to a text file and displays the content in the file content editor area of the "Checkpoint Properties Dialog Box" (described on page 1432) enabling you to configure your checkpoint. You can use parameters and regular expressions to augment your checkpoint, as needed.

You can perform a file content checkpoint for any of the following file types:

• HTML	• Microsoft Word	• Text
• PDF	• RTF	

If a file content checkpoint step fails during a run session, the Captured Data pane in the Run Results Viewer displays a side-by-side comparison of the generated document and the source document, enabling you to visually compare the differences between the documents, including lines or sections that were added or removed. For details, see the section on the section describing File Content Checkpoint Results (described in the *HP Run Results Viewer User Guide*).

Table Checkpoints Overview

Relevant for: GUI tests and scripted GUI components

You can use table checkpoints to check the content of tables displayed in your application. For example, you can check that a specified value is displayed in a certain cell. For some environments, you can also check the property values of the table object. For example, you can check that a table has the expected number of rows and columns.

During a run session, the table checkpoint compares the actual data to the expected data, as defined in the checkpoint. If the results match, the checkpoint passes. You can view the results of the checkpoint in the Run Results Viewer. For details, see the section on table checkpoint results in the *HP Run Results Viewer User Guide*.

Different environments support different checkpoints. For details on supported checkpoints, see "Supported Checkpoints" on page 2225.

Row Range Selection

The tables in your application may be very large. A table checkpoint on a large table may take a long time to create and a long time to run. You can choose to include all rows in your table checkpoint or you can specify a smaller row range.

For some UFT add-ins, when creating a new table checkpoint object, you can specify the range of rows you want to include using the "Define/Modify Row Range Dialog Box" on page 1467.

Page Checkpoints Overview

Relevant for: GUI tests and scripted GUI components

You can use page checkpoints to check statistical information about your Web pages. 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.

The following types of page checkpoints are available:

Individual Page Checkpoints

You can manually add a page checkpoint to check the links and image sources on a selected Web page during a recording or editing session.

For task details, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

For user interface details, see "Checkpoint Properties Dialog Box" on page 1432.

Automatic Page Checkpoints

You can instruct UFT to create automatic page checkpoints for every page during a recording session 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). By default, the automatic page checkpoint includes the checks that you select from among the available options in the Web > Advanced pane.

You can also instruct UFT not to perform automatic page checkpoints during run sessions by selecting the **Ignore automatic checkpoints while running tests** check box in the **Web > Advanced** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Web > Advanced** node).

For details, see the chapter on the **Web > Advanced** pane in the GUI Testing tab of the Options dialog box in the *HP Unified Functional Testing Add-ins Guide*.

Checking Text Overview

Relevant for: GUI tests and scripted GUI components

You can check that a specified text string is displayed by adding one of the following checkpoints.

- **Standard Checkpoint.** Enables you to check the text property of an object. You can use standard checkpoints to check text in Windows-based and other types of applications (including Web-based applications). For details on standard checkpoints, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

- **Text Area Checkpoint.** Enables you to check that a text string appears within a defined area in a Windows application, according to specified criteria. When checking text displayed in a Windows-based application, it is often advisable to define a text area larger than the actual text you want UFT to check. You then use the "Checkpoint Properties Dialog Box" (described on page 1432) to configure the relative position of the Checked Text within the captured area. During a run session, UFT checks for the selected text within the defined area, according to the settings you configured.

- **Text Checkpoint.** Enables you to check that the text is displayed in a screen, window, or Web page, according to specified criteria. For example, suppose you want to check the third occurrence of a particular text string in a page. To check for this string, you can specify which text precedes and/or follows it and to which occurrence of the specified text string you are referring.

By default, when checking text, UFT tries to retrieve the text directly from the object. If UFT cannot retrieve the text in this manner (for example, because the text is part of a picture), it tries to retrieve the text using an OCR (optical character recognition) mechanism. The OCR mechanism translates images of handwritten or typewritten text into machine-editable text.

> **Note:** The OCR text mechanism is not supported for objects in the Active Screen.

For task details, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

Text Recognition Overview

Relevant for: GUI tests and components

When working with tests and scripted components, you can use the text and text area checkpoint or output value commands to verify or retrieve text in your objects.

When working with tests, keyword or scripted components, and function libraries, you can insert steps to capture the text from objects in your application using the <*testobject*>**.GetVisibleText**, the *testobject***.GetTextLocation** test object methods, the **TextUtil.GetText** or **TextUtil.GetTextLocation** reserved object methods, or the *testobject***.GetText** (for Terminal Emulator objects).

When you use one of these options, UFT identifies text in your application using either a Windows API-based mechanism (not supported for Vista operating systems and higher, or any 64-bit operating systems) or an OCR (optical character recognition) mechanism.

By default, UFT tries to retrieve the text directly from the object using a Windows API-based mechanism. If UFT cannot capture the text this way (for example, because the text is part of a picture), it tries to capture the text using an OCR (optical character recognition) mechanism.

When UFT uses the OCR mechanism, a number of factors can affect the text it retrieves. Depending on the characteristics of the text you want to retrieve, you can adjust several OCR configuration options to optimize the way the text is captured.

You use the "Text Recognition Pane (Options Dialog Box > GUI Testing Tab)" (described on page 542) to specify the preferred text recognition mechanism and OCR-specific settings.

Guidelines for Text Recognition

Relevant for: GUI tests and components

- The Windows API text recognition mechanism is provided as is (not supported for Vista operating systems and higher, or 64-bit operating systems). If it does not work as you expect, change the option to use OCR.

  However, although UFT uses a market-leading OCR mechanism and the accuracy is usually very high, keep in mind that as with all OCR technologies, these engines are non-deterministic in the way they are built and interpret text and therefore accuracy of recognition for some text patterns may be better than others. The OCR mechanism does not guarantee 100% accuracy.

  Additionally, text-capturing steps may behave differently in different run sessions depending on the operating system version you are using, the installed service packs, other installed toolkits, the APIs used in your application, and so on.

- When using the OCR mechanism, the larger the text, the better the text recognition.

- The OCR mechanism is not supported for objects in the Active Screen.

- When working with text area checkpoints or specifying the text area in methods such as **GetTextLocation**, try to keep the dimensions of the selected text area as small as possible, as this helps prevent additional unwanted characters in recognized text.

  At the same time, consider the potential movement (change of coordinates) of the object within the window. For example, the screen resolution is often different on different computers, and this can affect the coordinates of the object in the application. Also, during the design and development stages of an application, an object may be moved to make room for other objects or for aesthetic purposes.

Consider that the operating system, installed service packs, installed toolkits, and so on, can all affect the size and location of an object in an application. Make sure that the dimensions of the selected text area are large enough for different system configurations.

The dimensions of the selected text area need to be large enough to take these issues into account.

- Single text block mode and multiple text block mode sometimes result in different captured text.

  If you are not sure which text block mode to use, use the default multiple block mode. If the results are not what you expect, then try using the single text block mode.

  For an example of when to use different text block modes, see "Checking Text in an Image - Use-Case Scenario" on page 1414.

  > **Tip:** If you want to use the text recognition mechanism for a large area containing different fonts and backgrounds, it is recommended to create several steps to capture the text for each single text block instead of creating one step to capture a multiple text block.

- Windows provides various themes. When working with text recognition, try to apply themes in the following order:

  - Windows Vista theme (for best results)
  - Windows XP theme
  - Windows Classic theme

  For example, this may be useful if the default OCR settings result in additional unexpected characters in the captured text.

- On Windows Vista operating systems and higher (including Windows 8), only the OCR mechanism is applied for text recognition. This can affect text recognition features (such as text checkpoints and output values, **GetVisibleText** and **GetTextLocation** test object methods, and **TextUtil.GetText** and **TextUtil.GetTextLocation** reserved object methods).

- If the text recognition mechanism retrieves unwanted text information (such as hidden text and shadowed text that appears as multiple copies of the same string) when using the multiple text block mode, use the single text block mode option. To do this, in the **Text Recognition** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Text Recognition** node), select **Single text block mode**. For details, see "Text Recognition Pane (Options Dialog Box > GUI Testing Tab)" on page 542.

- If your text recognition options are set to use the Windows API mechanism, then, when running a step that uses text recognition, the Windows API may cause a "blinking effect" in your application as it captures the text. If your test or component contains consecutive steps that utilize the text recognition mechanism, the "blinking effect" in one step may cause the subsequent text recognition step (or other step that relies on the appearance of the application, such as a bitmap checkpoint) to fail.

To address this, you can insert a **Wait** statement prior to each such step. This enables you to delay the performance of the next text recognition step until the Windows API capture of the previous step is complete.

- It is highly recommended to check text from your application window by inserting a standard checkpoint for the object containing the desired text, using its **text** (or similar) property.
- By default, when UFT captures text for a text/text area checkpoint or output value step using the **GetText**, **GetTextLocation**, or **GetVisibleText** methods, it tries to retrieve the text directly from the object using a Windows API-based mechanism. If UFT cannot capture the text this way (for example, because the text is part of a picture), it tries to capture the text using an OCR (optical character recognition) mechanism. For details about changing this behavior, see the *Can QuickTest Professional Text Recognition behavior be modified* Knowledge base article (number KM202721).
- **For tests:** If you are creating text area checkpoints, see the **Important Information** section in "Checkpoint Properties Dialog Box" on page 1432 for additional guidelines.

Text Recognition and Development Environments

Relevant for: GUI tests and components

The following table lists the development environments supported by UFT (via its add-ins) and specifies what is supported for text recognition.

Development Environment	Text Recognition Supported	Text Recognition Not Supported
ActiveX	Full text recognition support	N/A
Delphi	Full text recognition support	N/A
Java	• Text checkpoints • Text output values • Text area checkpoints • Text area output values	• **GetTextLocation** method • **GetVisibleText** method

Development Environment	Text Recognition Supported	Text Recognition Not Supported
.NET WebForms	• Text checkpoints for Page object only • Text output values for Page object only	• Text checkpoints for all other objects • Text output values for all other objects • Text area checkpoints for all objects • Text area output values for all objects • **GetTextLocation** method • **GetVisibleText** method
.NET WinForms	Full text recognition support	N/A
Oracle	N/A	No text recognition support
PeopleSoft	• Text checkpoints for PSFrame object only • Text output values for PSFrame object only	• Text checkpoints for all other objects • Text output values for all other objects • Text area checkpoints for all objects • Text area output values for all objects • **GetTextLocation** method • **GetVisibleText** method
PowerBuilder	Full text recognition support	N/A
SAP Gui for Windows	N/A	No text recognition support

Development Environment	Text Recognition Supported	Text Recognition Not Supported
SAP Web	• Text checkpoints • Text output values	• Text area checkpoints • Text area output values • **GetTextLocation** method • **GetVisibleText** method
Siebel	N/A	No text recognition support
Silverlight	Full support except as listed in the **Not Supported** column.	• **GetTextLocation** method
Standard Windows	Full text recognition support	N/A
Stingray	Full text recognition support	N/A
Terminal Emulators	Text output values for TeScreen and TeTextScreen objects only	• Other text checkpoints • Other text output values • Text area checkpoints • Text area output values • **GetTextLocation** method • **GetVisibleText** method
VisualAge	Full text recognition support	N/A
Visual Basic	Full text recognition support	N/A

Development Environment	Text Recognition Supported	Text Recognition Not Supported
Web	- Text checkpoints for Page object only - Text output values for Page object only	- Text checkpoints for all other objects - Text output values for all other objects - Text area checkpoints for all objects - Text area output values for all objects - **GetTextLocation** method - **GetVisibleText** method
WPF	Full support except as listed in the **Not Supported** column.	- **GetTextLocation** method

Checking Text in an Image - Use-Case Scenario

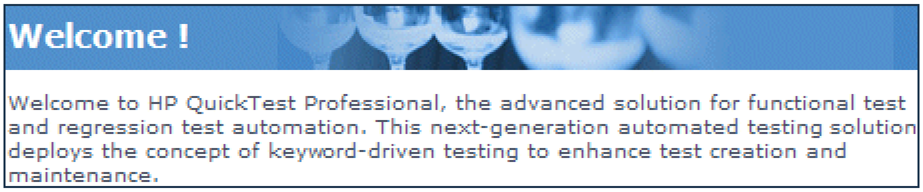
Relevant for: GUI tests and scripted GUI components

Ben and George are quality assurance engineers who are experienced UFT users. George is also familiar with text recognition and has a basic understanding of how text recognition mechanisms work.

Ben often uses bitmap checkpoints to check the appearance of various icons or pictures in the user interface he is testing.

For one of his projects, Ben also needed to verify the text in the graphics, so he decided to use text checkpoints.

Ben decided to begin the verification process by inserting a text checkpoint to check that the text Welcome ! was displayed correctly in the following graphic.

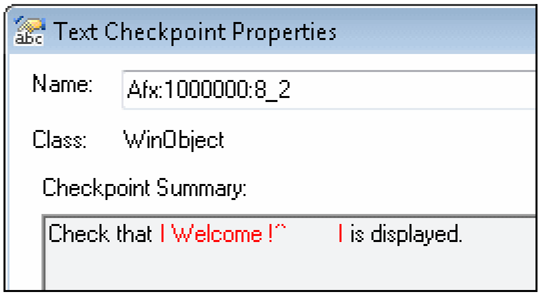


Welcome !

Welcome to HP QuickTest Professional, the advanced solution for functional test and regression test automation. This next-generation automated testing solution deploys the concept of keyword-driven testing to enhance test creation and maintenance.

Before inserting the text checkpoint, Ben opened the Text Recognition pane and configured the text recognition settings. Ben set the text recognition mechanism to **Use Only OCR** because the text was part of a graphic. Ben also knew that single text block mode usually works best, so he selected the **Single text block mode** option.

Ben then inserted a text checkpoint on the entire area shown above and received the following results in the Text Checkpoint Properties dialog box:



Text Checkpoint Properties

Name: Afx:1000000:8_2

Class: WinObject

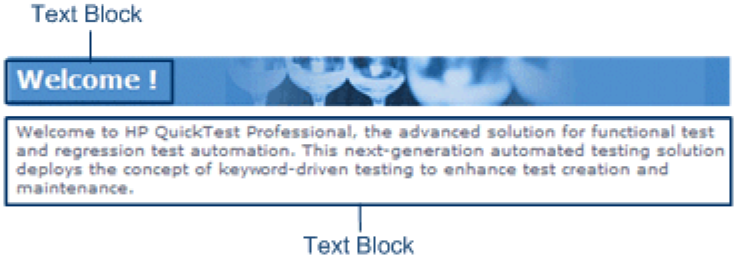
Checkpoint Summary:

Check that | Welcome !^ | is displayed.

Ben noticed that there were extra characters in the **Checkpoint Summary** area of the text checkpoint, but he did not know why.

Ben asked his colleague, George, for help. George explained to him that the text recognition mechanism sometimes adds extra characters to the text checkpoint when it does not recognize the text correctly.

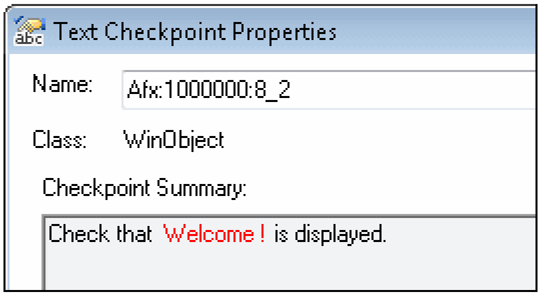
George also pointed out that the area Ben defined for the text checkpoint consisted of multiple text blocks because the text was not uniform in font size, color, or background. The title area consisted of white characters on a blue-gray background, while the remaining text was smaller and consisted of blue text on a white background.



Welcome !

Welcome to HP QuickTest Professional, the advanced solution for functional test and regression test automation. This next-generation automated testing solution deploys the concept of keyword-driven testing to enhance test creation and maintenance.

Ben remembered that he had selected the **Single text block mode** option in the **Text Recognition** pane (**Tools > Options > GUI Testing** tab **> Text Recognition** node) and understood that if he wanted to use single text block mode, he would have to create a text checkpoint only on the Welcome ! area of the graphic, and not on the entire graphic. Ben tried this, and the OCR mechanism correctly identified the text, as shown below:



Text Checkpoint Properties

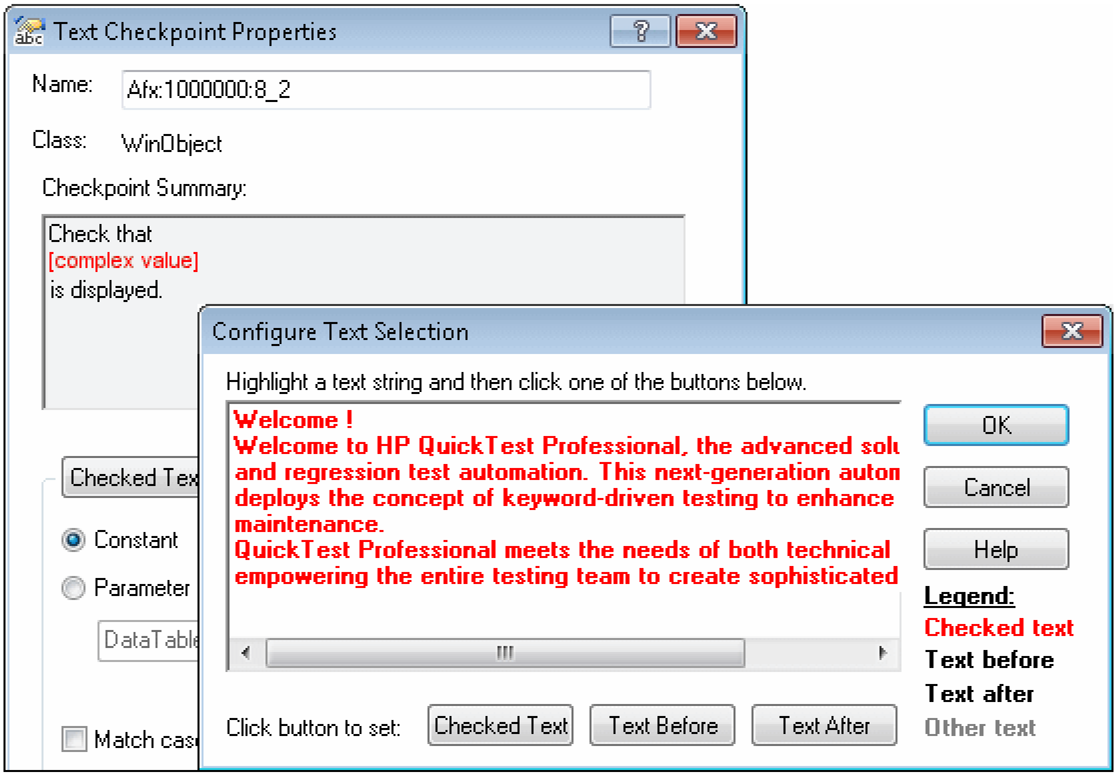
Name: Afx:1000000:8_2

Class: WinObject

Checkpoint Summary:

Check that Welcome ! is displayed.

Ben was pleased with the results, but he wanted to explore other possibilities, so he inserted another text checkpoint—this time on the entire graphic. He selected the **Multiple text block mode** option in the Text Recognition pane, which resulted in the following:



Text Checkpoint Properties

Name: Afx:1000000:8_2

Class: WinObject

Checkpoint Summary:

Check that
[complex value]
is displayed.

Configure Text Selection

Highlight a text string and then click one of the buttons below.

Welcome !
Welcome to HP QuickTest Professional, the advanced solı
and regression test automation. This next-generation auton
deploys the concept of keyword-driven testing to enhance
maintenance.
QuickTest Professional meets the needs of both technical
empowering the entire testing team to create sophisticated

Legend:
Checked text
Text before
Text after
Other text

Click button to set:

Ben was pleased that the OCR mechanism correctly recognized all of the text in the graphic. But he needed to check only the title, Welcome !, so he finalized this checkpoint by marking all of the text after Welcome ! as **Text After**.

Even though both checkpoints passed, Ben needed only one text checkpoint. He decided to keep the first checkpoint (that used **Single text block mode**), and he deleted the second one. He selected the **Single text block mode** option in the Text Recognition pane to help ensure that the checkpoint would pass in future run sessions.

XML Checkpoints Overview

Relevant for: GUI tests and scripted GUI components

You can use XML checkpoints to check the contents of individual XML data files or documents that are part of your Web application.

XML (Extensible Markup Language) is a meta-markup language for text documents that is endorsed as a standard by the W3C (World Wide Web Consortium). XML makes the complex data structures portable between different computer environments/operating systems and programming languages, facilitating the sharing of data.

XML files contain text with simple tags that describe the data within an XML document. These tags describe the data content, but not the presentation of the data. Applications that display an XML document or file use either CSS (Cascading Style Sheets) or XSL-FO (XSL Formatting Objects) to present the data.

You can perform checkpoints on XML documents contained in Web pages or frames, on XML files, and on test objects that support XML (such as **WebXML** test objects). An XML checkpoint is a verification point that compares a current value for a specified XML element, attribute and/or value with its expected value. When you insert a checkpoint, UFT adds a checkpoint step in the Keyword View and adds a Check CheckPoint statement in the Editor. During a run session, UFT compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails.

After a run session, you can view summary results of the XML checkpoint in the Run Results Viewer. You can also view detailed results by opening the XML Checkpoint Results window, described in the *HP Run Results Viewer User Guide*.

A few common uses of XML checkpoints are described below:

- An XML file can be a static data file that is accessed to retrieve commonly used data for which a quick response time is needed—for example, country names, zip codes, or area codes. Although this data can change over time, it is normally quite static. You can use an XML file checkpoint to validate that the data has not changed from one application release to another.

- An XML file can consist of elements with attributes and values (character data). There is a parent and child relationship between the elements, and elements can have attributes associated with them. If any part of this hierarchy (including data) changes, your application's ability to process the XML file may be affected. Using an XML checkpoint, you can check the content of an element to make sure that its tags, attributes, and values have not changed.

- XML files are often an intermediary that retrieves dynamically changing data from one system. The data is then accessed by another system using Document Type Definitions (DTD), enabling the accessing system to read and display the information in the file. You can use an XML checkpoint and parameterize the captured data values if you want to check an XML document or file whose data changes in a predictable way.

- XML documents and files often need a well-defined structure to be portable across platforms and development systems. One way to accomplish this is by developing an XML schema, which describes the structure of the XML elements and data types. You can use schema validation to check that each item of content in an XML file adheres to the schema description of the element in which the content is to be placed.

> **Note:** XML checkpoints are compatible with namespace standards. A change in namespace between expected and actual values results in a failed checkpoint.
>
> For details on XML standards, see: http://www.w3.org/XML/
>
> For details on namespace standards, see: http://www.w3.org/TR/1999/REC-xml-names-19990114/

XML Checkpoint Types

Relevant for: GUI tests and scripted GUI components

You can create the following types of XML checkpoints:

- **XML Web Page/Frame Checkpoint.** Checks an XML document within a Web page or frame.
- **XML File Checkpoint.** Checks a specified XML file.

Using XML Objects and Methods to Enhance Your Test or Scripted Component

Relevant for: GUI tests and scripted GUI components

UFT provides several scripting methods that you can use with XML data. You can use these scripting methods to retrieve data and return new XML objects from existing XML data. You do this by using the XMLUtil, or WebXML objects to return XML data and then using the supported XMLData objects and methods to manipulate the returned data.

> **Tip:** All XMLData objects and methods are compatible with namespace and XPath standards.
>
> For details on XML standards, see: http://www.w3.org/XML/
>
> For details on namespace standards, see: http://www.w3.org/TR/1999/REC-xml-names-19990114/
>
> For details on XPath standards, see: http://www.w3.org/TR/1999/REC-xpath-19991116

For details on programming in the Editor, see "Programming in GUI Testing Documents in the Editor" on page 994. For details on XML objects and methods, see the **Supplemental Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

Tasks

How to Insert a Checkpoint in a GUI Test or Component

Relevant for: GUI tests and components

This task describes how to insert a checkpoint step while recording or editing your test or component. You can also add an existing checkpoint to a test or scripted component. It is generally more convenient to define checkpoints after creating the initial test or component.

For details on supported checkpoints per add-in environment, see "GUI Checkpoints and Output Values Per Add-in" on page 2224

This task describes both the general process for inserting a new checkpoint step to your test or component and the prerequisites and considerations for the different types of components.

This task includes the following steps:

- "Important information before inserting the checkpoint" on the next page
- "Define automatic page checkpoints - optional" on page 1421
- "Set global accessibility checkpoint preferences" on page 1421
- "Insert a checkpoint step while recording your test or component" on page 1422
- "Insert a checkpoint step while editing your test or component" on page 1422
- "Use programming to insert checkpoints in a test or scripted component" on page 1424
- "Set options for the checkpoint" on page 1424
- "Move checkpoint objects from the local object repository to a shared object repository - optional" on page 1426

Important information before inserting the checkpoint

Object visibility in application	• During an editing session, make sure the object is visible in your application before inserting a standard checkpoint. • **For bitmap checkpoints:** During a run session, bitmap checkpoints can capture only the visible part of an object. Therefore, confirm that the object to capture is always fully visible on the screen before a bitmap checkpoint step is performed. One way to do this is to insert a **MakeVisible** statement (for relevant environments) prior to your bitmap checkpoint step. For details on the **MakeVisible** method, see the *HP UFT Object Model Reference for GUI Testing*. • **For file content checkpoints:** The source file must be located on the file system.
Availability	• Recording sessions • Editing sessions • Active Screen (not supported for file content checkpoints or XML checkpoints)

Important Information	• Checkpoints can be viewed in the following modes: ▪ **Simple Mode:** Displays only the basic properties and expected values of the checkpoint. ▪ **Advanced Mode:** Displays all supported properties and expected values of the checkpoint. • In ALM, you cannot create, edit, or rename checkpoints for keyword GUI components. • You cannot create image, table, or (Web) page checkpoints in a keyword GUI component. These special checkpoint types are only available for tests and scripted GUI components. However, if you select a Web page or any table object when creating a standard checkpoint for your component, you can check their object properties like any other object. • **For bitmap checkpoints:** If you want to create a bitmap checkpoint that contains multiple objects, you should select the highest level object that includes all the objects to include in the bitmap checkpoint. • **For text or text area checkpoints:** ▪ Before you create a text or text area checkpoint for a Windows-based application, make sure you configure the required capture settings in the Text Recognition pane (**Tools > Options > GUI Testing** tab > **Text Recognition** node. For details, see "Text Recognition Pane (Options Dialog Box > GUI Testing Tab)" on page 542. ▪ You can also check the text property of an object in Windows-based and other types of applications (including Web-based applications) by using a standard checkpoint. • **For XML checkpoints:** You can insert XML checkpoints to verify Web pages and frames and to directly access and verify specific XML files in your system.

Define automatic page checkpoints - optional

In the **Web > Advanced** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Web > Advanced** node), do one or more of the following:

- To instruct UFT to create automatic page checkpoints for every page during every recording session, select the **Create a checkpoint for each Web page while recording** check box.

- To instruct UFT not to perform automatic page checkpoints during run sessions, select the **Ignore checkpoints while running tests** check box.

Set global accessibility checkpoint preferences

In the Web Advanced pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Web > Advanced** node), do one or more of the following:

- Define the checks to include in the checkpoints. All accessibility checkpoints in your test use the options that are selected in the Advanced Web Options dialog box at the time of the run session. You can also view these options in the Accessibility Checkpoint Properties dialog box.
- **(Optional)** To instruct UFT to insert an accessibility checkpoint for each page as you record, select the **Add Automatic accessibility checkpoint to each Web page while recording** check box. This creates an accessibility checkpoint for each page as you record.

Insert a checkpoint step while recording your test or component

1. Start a recording session before inserting a checkpoint.
2. Insert a checkpoint by doing one of the following:
   - In the Record toolbar, click the **Insert Checkpoint or Output Value** button and select the type of checkpoint from the drop-down list.
   - Select **Design > Checkpoint** and choose the relevant type of checkpoint.
   - Click the **Insert Checkpoint or Output Value** button in the toolbar and select the type of checkpoint from the drop-down list.
3. UFT is hidden, and the pointer changes to a pointing hand. In your application, click the object that you want to check.

   > **Note:** If the object in your application is associated with more than one location, the Object Selection dialog box opens. This dialog box enables you to select an object to check from the object tree. The objects in the tree are displayed with hierarchical order, based on the location you clicked in the Active Screen or application.

Insert a checkpoint step while editing your test or component

1. You may need to open the application and display the relevant object before inserting a checkpoint. This depends on the environment and the object type you are checking. For details, see the prerequisite information for your specific checkpoint type.
2. Select the step where you want to add the checkpoint and do one of the following:
   - Select **Design > Checkpoint**, and then select the relevant checkpoint option.
   - Select **Design > Checkpoint > Existing Checkpoint**.
   - Right-click any object in the Active screen and select the relevant checkpoint. You can create checkpoints for any object in the Active Screen even if the object is not part of any step in the Keyword View.

If you use the Active Screen to insert a checkpoint, ensure that the Active Screen contains sufficient data for the object you want to check. For details, see the section on the "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

> **Note:** If the object in your application is associated with more than one location, the Object Select Dialog Box opens. This dialog box enables you to select an object to check from the object tree. The objects in the tree are displayed in hierarchical order, based on the location you clicked in the Active Screen or application.

Notes:

- **For table checkpoints:** When inserting a table checkpoint, for certain objects in certain environments, before the Table Checkpoint Properties dialog box opens, the Define/Modify Row Range Dialog Box opens. In this dialog, select the row range to check.

- **For text or text area checkpoints:**

  - To create the checkpoint, you first highlight a text string in the Active Screen then right-click the string, and select **Insert Text Checkpoint**.

  - When you create a text area checkpoint, you first define the area containing the text you want UFT to check.

    When you select the Text Area Checkpoint option, the mouse turns into a crosshairs pointer. Click and drag the crosshairs point to define this area. Release the mouse button after outlining the area required.

    > **Tip:** Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

- **For file content checkpoints:** When inserting a file content checkpoint, the File Content Checkpoint Properties dialog box displays by default the option to select only **All Supported Files**. When this is selected, only files with the expected extensions are displayed (for example, .htm or .pdf files). You can also select a file that uses a non-standard extension by selecting **All Files** in the **Files of type** box and then selecting the relevant file.

- **For database checkpoints:**

  When inserting a database checkpoint, the Database Query Wizard opens.

  a. In the Database Query Wizard, define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement.

  b. If you selected Microsoft Query as your data source, Microsoft Query opens, enabling you to define a query. When you are done, in the Finish page of the Query Wizard, use one of the following:

- **Exit and return to HP Unified Functional Testing.** Exits Microsoft Query.
- **View data or edit query in Microsoft Query.** View or edit the query prior to exiting Microsoft Query.

c. If you selected **Specify SQL statement manually**, the Specify SQL statement page opens, enabling you to specify the connection string and the SQL statement.

Use programming to insert checkpoints in a test or scripted component

- If you want to retrieve the return value of a checkpoint (a boolean value that indicates whether the checkpoint passed or failed), you must add parentheses around the checkpoint argument in the statement in the Editor. For example:

```
a = Browser("MyBrowser").Page("MyPage").Check (CheckPoint("MyProperty"))
```

- You can also use the **CheckProperty** method and the **CheckItemProperty** method to check specific property or item property values. For details, see the *HP UFT Object Model Reference for GUI Testing*.

Set options for the checkpoint

In the "Checkpoint Properties Dialog Box", specify the settings for the checkpoint object.

For table checkpoints:

Define the cell selection for the table object in the Grid area of Table Checkpoint Properties dialog box, as follows:

To:	Do this:
Add a **single cell** to or remove it from the check	Double-click the cell
Add an **entire row** to or remove it from the check	Double-click the row header
Add an **entire column** to or remove it from the check	Double-click the column header.
Add **all cells** to or remove all cells from the check	Double-click the column header.

Add **a range of cells** to the check	Select the cells to add to the check and click the **Add to Check** button
Remove **a range of cells** from the check	Select the cells to remove from the check and click the **Remove from Check** button

For database checkpoints:

Define the cell selection for the database object in the Grid area of Database Checkpoint Properties dialog box, as follows:

To:	**Do this:**
Add a **single cell** to or remove it from the check	Double-click the cell
Add an **entire row** to or remove it from the check	Double-click the row header
Add an **entire column** to or remove it from the check	Double-click the column header.
Add **all cells** to or remove all cells from the check	Double-click the column header.
Add **a range of cells** to the check	Select the cells to add to the check and click the **Add to Check** button
Remove **a range of cells** from the check	Select the cells to remove from the check and click the **Remove from Check** button

To modify the SQL query definition, in the Keyword View or Editor, right-click the database object that you want to modify and select **Object Properties**.

For file content checkpoints:

In the File Content Checkpoint Properties dialog box, scroll to each line you want to compare and select it.

As you hover over a line, a checkbox and a regular expression icon are displayed in the sidebar to the left of that line.

- Click the check box to select (or clear) the line for verification.
- Click the Treat Line as Regular Expression/Plain Text button to add (or remove) backslashes prior to all special characters in that line. You can then modify any regular

expressions, as needed.

Note:

- If the source file contains multiple ages, the File Content Editor is divided into separate pages. You can then expand or collapse the pages, select or clear entire pages for verification and so on.
- Pages with more than 19 MB of content may take a long time to parse.

Move checkpoint objects from the local object repository to a shared object repository - optional

After you insert a checkpoint step, the checkpoint object is added to the local object repository. If you are using shared object repositories, you can move the new checkpoint object to your shared object repository.

How to Include and Ignore Areas When Comparing a Bitmap - Use-Case Scenario

Relevant for: GUI tests and components

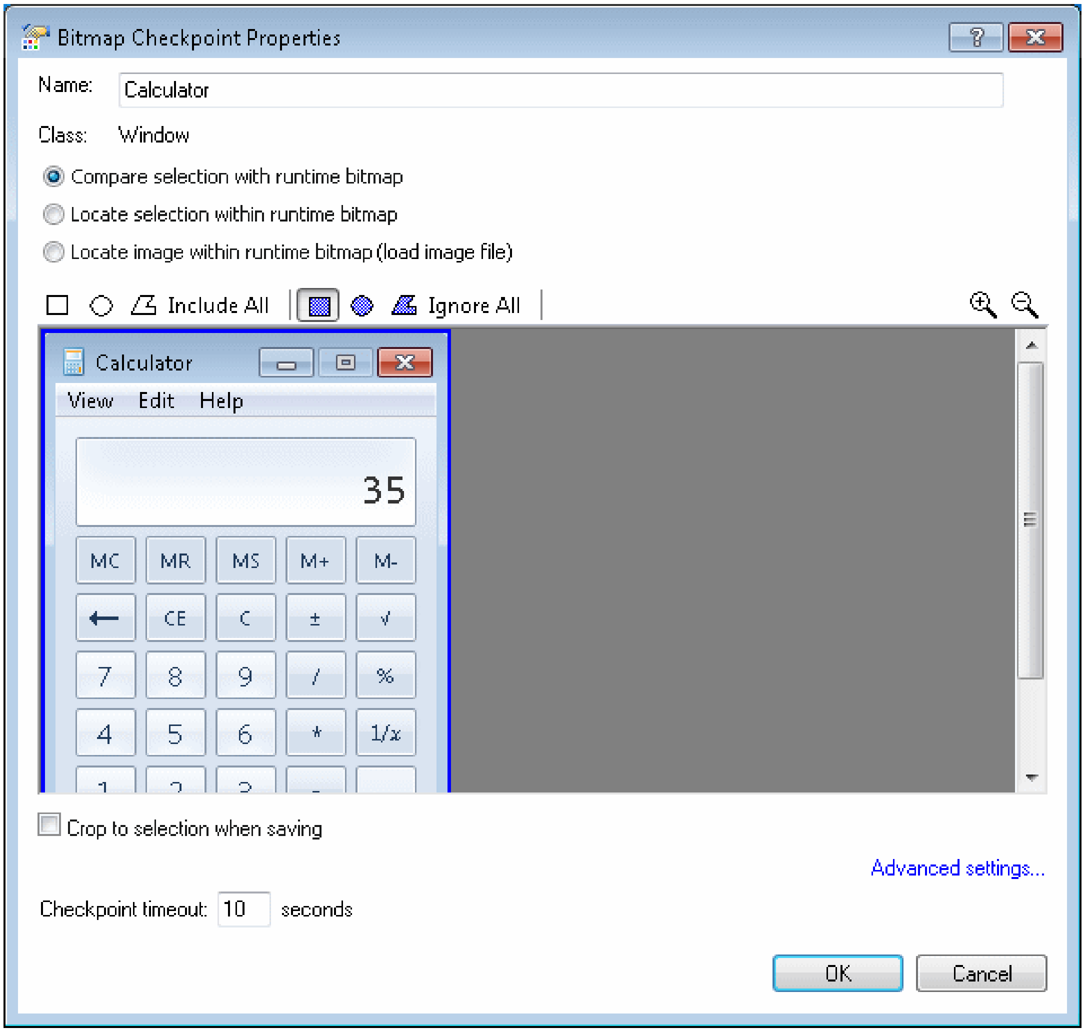
Suppose you want to compare an expected bitmap with an actual bitmap of the object in your application, but there are areas of this runtime bitmap that might be affected by dynamic values and parameters. If you include these areas in the checkpoint, running the steps with different values may cause the checkpoint to fail.

This use-case scenario describes the process you would follow to define which areas to ignore and include in a bitmap checkpoint that compares an expected bitmap with a runtime bitmap.

Note: For a task related to this scenario, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419.

1. In UFT, start a recording session and open the Windows Calculator application (for example, **Start > Programs > Accessories > Calculator**).
2. Record steps that multiply **7** by **5** and display the result.
3. Select **Design > Checkpoint** menu, or click the **Insert Checkpoint or Output Value** button in the toolbar, and then select **Bitmap Checkpoint**. UFT is hidden, and the pointer changes to a pointing hand.

4. Select the Calculator application. If the "Object Selection Dialog Box " (described on page 1195) opens, select the top-level object and click **OK**. The "Checkpoint Properties Dialog Box" (described on page 1432) opens.



Bitmap Checkpoint Properties

Name: Calculator

Class: Window

● Compare selection with runtime bitmap

○ Locate selection within runtime bitmap

○ Locate image within runtime bitmap (load image file)

□ Crop to selection when saving

Advanced settings...

Checkpoint timeout: 10 seconds

OK Cancel

> **Note:** (For tests and scripted components) For the purpose of this use-case scenario, you insert the bitmap checkpoint while recording your steps. However, the options are relevant also when inserting a bitmap checkpoint from the Active Screen.

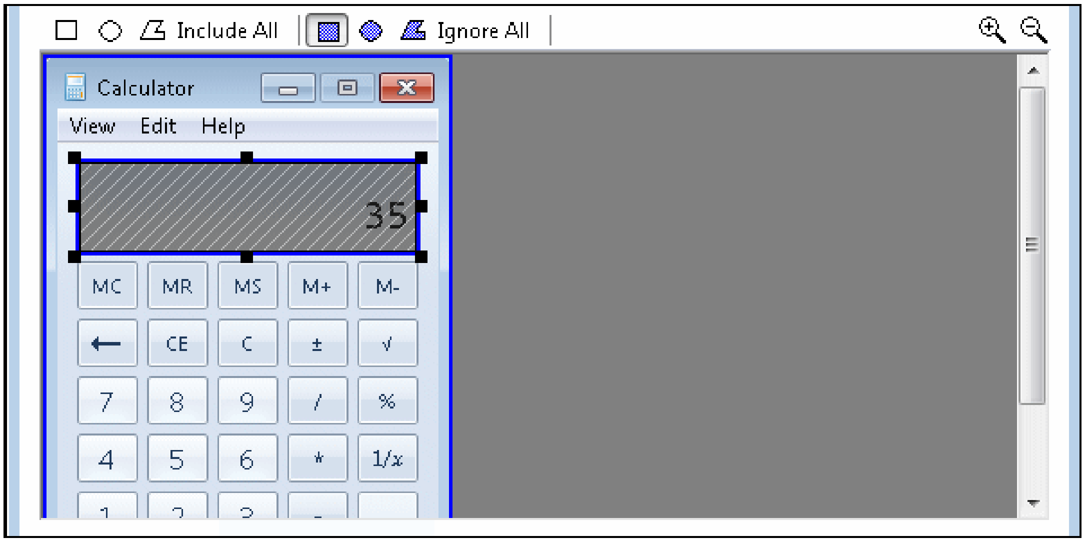
5. In the Bitmap Checkpoint Properties dialog box, select the **Compare selection with runtime bitmap** radio button. For the purpose of this scenario, we will compare a bitmap of the entire Calculator application.

6. Because the value in the number line may change, depending on parameters that are used during the run session, we want to ignore the number line when comparing the expected bitmap

with the runtime bitmap.

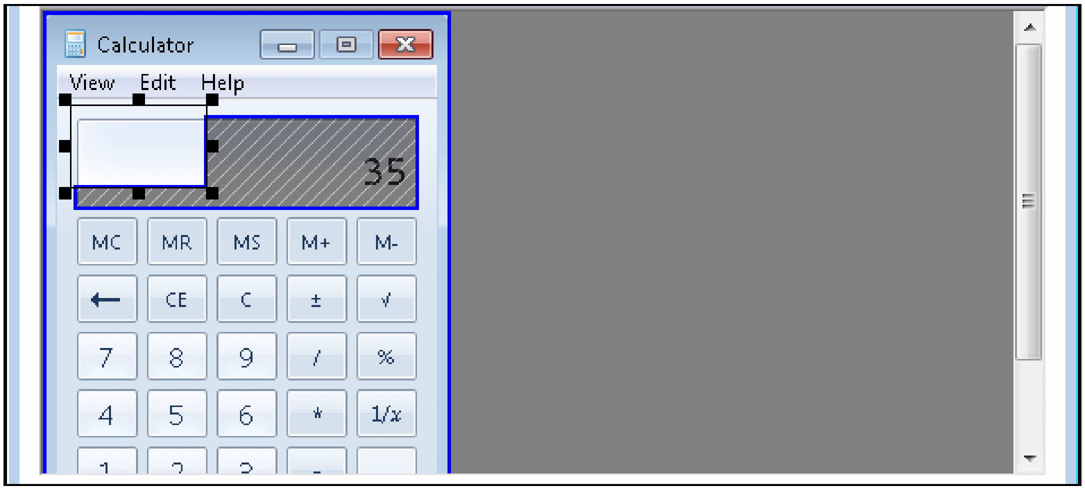
To do this, click the **Mark Rectangle to Ignore** button ▨ in the toolbar, and then select the entire number line. Modify the size and position of your selection as needed. When you are finished, double-click to finalize the selection.

The following example shows the bitmap display area when the number line is selected to be ignored:

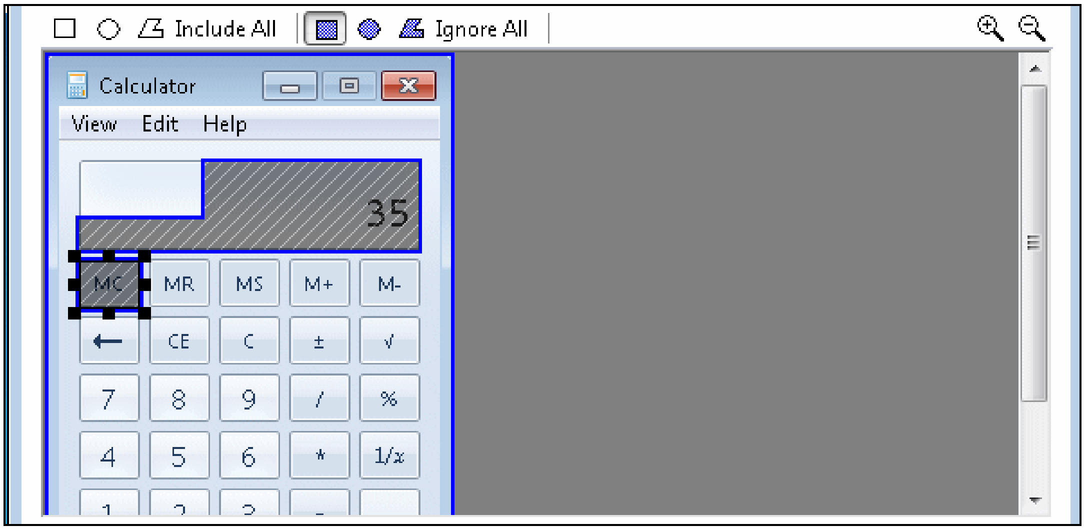


7. (Optional) If you need to modify your ignored selection after you finalize it, you can do so by clicking the **Mark Rectangle to Include** button □ in the toolbar, and then selecting the areas that you want to clear. Any areas that you clear from the ignored selection are included in the comparison.

The following example shows the bitmap display area after a part of the ignored selection is selected for clearing (double-click the area to finalize):



8. (Optional) If you want additional areas to be ignored, click the **Mark Rectangle to Ignore** button ▦ , select the relevant areas, and double-click to finalize.



9. Click **OK** to close the Bitmap Checkpoint Properties dialog box and insert the bitmap checkpoint. When you run your steps, the bitmap checkpoint ignores the area that you selected, and the checkpoint passes even if the value in the number line changes.

How to Configure Text Recognition Settings

Relevant for: GUI tests and components

This task describes a suggested workflow for configuring text recognition settings.

This task includes the following steps:

- "Prerequisites" below
- "Analyze the characteristics of the text" below
- "Set the appropriate options in the Text Recognition Pane " below
- "Check the text recognition settings" below
- "Adjust the settings as necessary" on the next page
- "Results" on the next page

1. **Prerequisites**

   In your application, display the text you want to capture.

2. **Analyze the characteristics of the text**

   Determine whether you can capture the text using a text (or text-like) property instead of using a text recognition mechanism.

   If it must use text-recognition, analyze whether the Windows API or OCR mechanisms are more likely to meet your needs.

   For details, see "Text Recognition Overview" on page 1408 and "Guidelines for Text Recognition" on page 1409.

3. **Set the appropriate options in the Text Recognition Pane**

   For details, see "Text Recognition Pane (Options Dialog Box > GUI Testing Tab)" on page 542.

4. **Check the text recognition settings**

   a. Create or open a test or component.

   b. Do any of the following:

      - Insert a text checkpoint or output value step (tests and scripted components only)
      - Insert a step that uses one of the following test object methods:
        - *testobject*.**GetVisibleText**
        - *testobject*.**GetTextLocation**
        - *testobject*.**GetText** (for Terminal Emulator objects)

- Insert a step that uses one of the following reserved object methods (tests and scripted components only):
    - *TextUtil*.**GetText**
    - *TextUtil*.**GetTextLocation**

c. Run the step to verify that text recognition works as expected.

For details, see "How to Insert a Checkpoint in a GUI Test or Component" on page 1419, "How to Create or Modify an Output Value Step" on page 1494, and the *HP UFT Object Model Reference for GUI Testing*.

5. Adjust the settings as necessary

If the captured text is not as expected, analyze the problems and adjust the Text Recognition options to fine tune the way UFT captures your text.

For details, see "Text Recognition Pane (Options Dialog Box > GUI Testing Tab)" on page 542 and "Guidelines for Text Recognition" on page 1409.

6. Results

You can now use text recognition to capture text from your application.

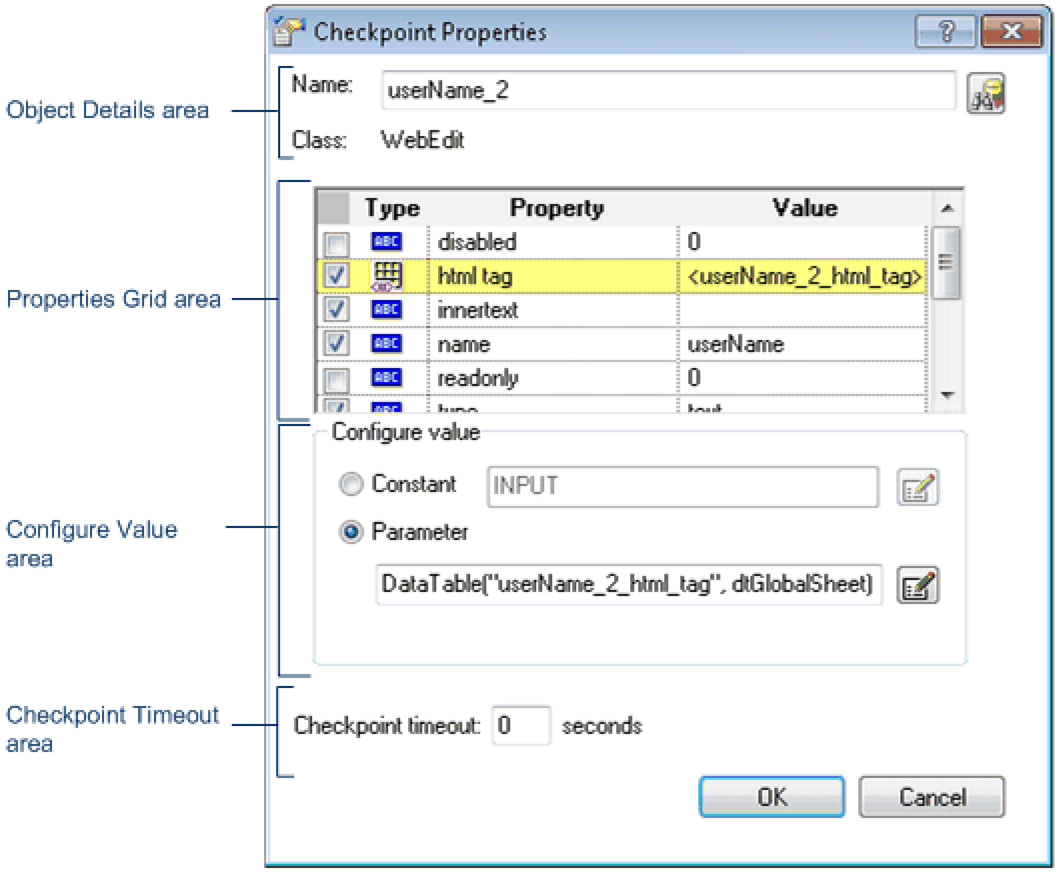
Reference

Checkpoint Properties Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to edit your checkpoint properties for a selected checkpoint object.

The example below gives the basic structure for the various Checkpoint Properties dialog boxes. Certain elements may differ between checkpoint types.



Name: userName_2

Class: WebEdit

	Type	Property	Value
□	ABC	disabled	0
☑		html tag	<userName_2_html_tag>
☑	ABC	innertext	
☑	ABC	name	userName
□	ABC	readonly	0
☑	ABC	type	text

Configure value

○ Constant INPUT

● Parameter

DataTable("userName_2_html_tag", dtGlobalSheet)

Checkpoint timeout: 0 seconds

To access	1. Ensure that a GUI action or component is in focus in the document pane. 2. Do one of the following: ▪ Insert a new checkpoint step and select an object from your application. ▪ Click in the Value cell of an existing checkpoint step, and click the checkpoint properties icon ☑ . ▪ **In an action or a scripted component:** Right-click an existing checkpoint step and select **Checkpoint Properties**. ▪ In the local or shared object repository, click an existing checkpoint object. The **Checkpoint** details are displayed on the right side of the object repository window, in the **Object Details** area.
Important information	The advanced properties of standard checkpoint objects cannot be viewed or edited in ALM. Therefore, if one or more advanced properties are selected in a standard checkpoint object, and an ALM user views its properties in ALM, the dialog box displays text indicating that some properties are not shown.

This dialog box has the following areas:

- "Object Details Area" below
- "Properties Grid Area" on the next page
- "Configure Value Area" on the next page
- "Statement Location and Checkpoint Timeout Area" on page 1435

Object Details Area

UI Element	Description
Name	The name that UFT assigns to the checkpoint object. By default, the name is the same as the name of the object on which the checkpoint step is being performed. You can specify a different name for the checkpoint object or accept the default name.
Class	The type of object (read-only). **Note for keyword components:** The class of the object is not displayed in ALM.

UI Element	Description
🔍	**Find in Repository.** Displays the checkpoint object in its object repository. This option is only available when editing an existing checkpoint step. **Note for keyword components:** This option only available in advanced mode.
⇊ ⇈ **(only available for Keyword components**	**Advanced Mode/Simple Mode.** Toggles the dialog box between Advanced Mode and Simple Mode. For details on Simple and Advanced mode, see "Simple and Advanced Mode" on page 1397.

Properties Grid Area

The Properties grid area displays the properties to check in the object.

For **standard checkpoints** see "Properties Grid Area (Checkpoint Properties Dialog Box) - Standard, Page, and Image Checkpoints" on the next page.

For **bitmap checkpoints**, see "Bitmap Options Section (Checkpoint Properties Dialog Box) - Bitmap Checkpoints" on page 1438.

For **table checkpoints** see "Properties Grid Area (Checkpoint Properties Dialog Box) - Table Checkpoints" on page 1446.

For **text or text area checkpoints**, see "Checkpoint Summary Area (Checkpoint Properties Dialog Box) - Text/Text Area Checkpoints" on page 1451.

For **file content checkpoints**, see "File Content Editor (Checkpoint Properties Dialog Box) - File Content Checkpoints" on page 1442.

For **database checkpoints**, see "Properties Grid Area (Checkpoint Properties Dialog Box) - Database Checkpoints" on page 1441.

For **page checkpoints**, see "Properties Grid Area (Checkpoint Properties Dialog Box) - Standard, Page, and Image Checkpoints" on the next page.

For **accessibility checkpoints**, see "Current Settings Area (Checkpoint Properties Dialog Box) - Accessibility Checkpoints" on page 1438.

For **XML checkpoints**, see "XML Tree/Options Area (Checkpoint Properties Dialog Box) - XML Checkpoints" on page 1457.

Configure Value Area

The Configure Value area displays the options to configure the value of the object to be checked.

For a **standard checkpoint**, see "Configure Value Area (Checkpoint Properties Dialog Box) - Standard/Image Checkpoints" on page 1436.

For a **bitmap**, **file content**, **accessibility**, or **XML checkpoint**, this area is not applicable.

For a **table** or **database checkpoint**, see "Configure Value Area (Checkpoint Properties Dialog Box) - Table/Database Checkpoints" on page 1448.

For a **text** or **text area checkpoint**, see "Checkpoint Options Area (Checkpoint Properties Dialog Box) - Text/Text Area Checkpoints" on page 1452.

For a **page checkpoint**, see "Configure Value Area (Checkpoint Properties Dialog Box) - Page Checkpoints" on page 1455.

Statement Location and Checkpoint Timeout Area

> **Note:** If you are working with a keyword GUI component, this option is available only in Advanced Mode.

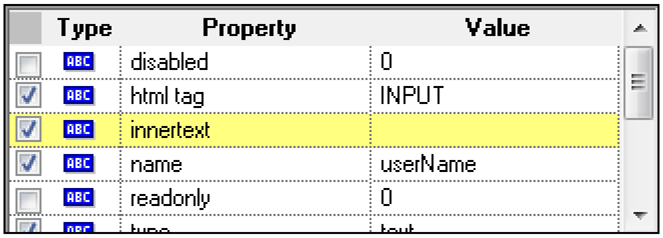
User interface elements are described below:

UI Element	Description
Checkpoint timeout	Specifies the time interval (in seconds) during which UFT attempts to perform the checkpoint successfully. UFT continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails. **Example:** Suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached. If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Run Results Viewer displays information on the checkpoint timeout. **Note:** The maximum allowed timeout for a checkpoint is 99999 seconds.
Insert statement	Specifies whether to insert the checkpoint step before or after the currently selected step. The default value is **Before current step**. **Note:** Available only when inserting a new checkpoint step during an editing session. During a recording session, the checkpoint step is always inserted as the next step.

Properties Grid Area (Checkpoint Properties Dialog Box) - Standard, Page, and Image Checkpoints

Relevant for: GUI tests and components

Specific properties may vary depending on the type of object you are checking.



	Type	Property	Value
□	ABC	disabled	0
☑	ABC	html tag	INPUT
☑	ABC	innertext	
☑	ABC	name	userName
□	ABC	readonly	0
☑	ABC	type	text

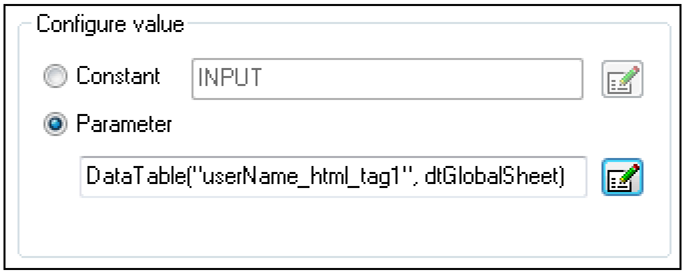
User interface elements are described below:

UI Elements	Description
Check box	For each object class, UFT recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.
Type	• ABC . Indicates that the value of the property is currently a constant. • . Indicates that the value of the property is currently a test or action parameter (tests only). • . Indicates that the value of the property is currently a Data Table parameter. (tests and scripted components only) • . Indicates that the value of the property is currently an environment variable parameter. (tests and scripted components only) • . Indicates that the value of the property is currently a random number parameter. (tests and scripted components only) • . Indicates that the value of the property is currently a parameter. (keyword components only)
Property	The name of the property.
Value	The expected value of the property. For details on modifying property values, see "Configure Value Area" on page 1434. **For keyword components:** See also: "Parameterization / Properties Dialog Box (Checkpoints)" on page 1473.

Configure Value Area (Checkpoint Properties Dialog Box) - Standard/Image Checkpoints

Relevant for: GUI tests and components

This area enables you to configure object property values or the values of the operation arguments defined for the step.



Configure value

○ Constant INPUT

● Parameter

DataTable("userName_html_tag1", dtGlobalSheet)

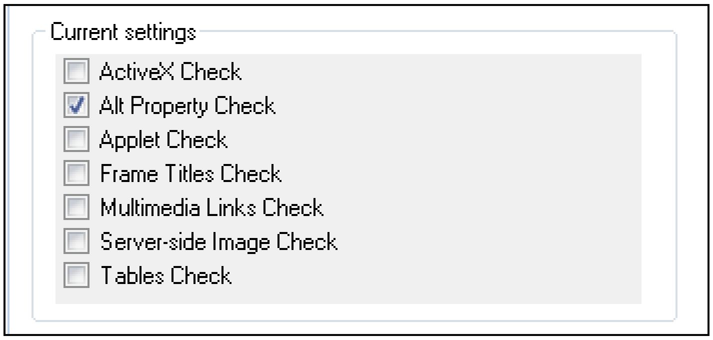
User interface elements are described below:

UI Elements	Description
Constant	A manually defined value that remains unchanged for the duration of the run. In certain contexts, you can define a constant value using a regular expression. If the entire value cannot be displayed in the **Constant** box, it is shown as **[complex value]**. For example, the value of a list's **all items** property is a multi-line value, where each line contains the value of an item in the list.
Parameter	A value that is defined or generated externally and is retrieved during a run session. For example, a parameter value may be defined in an external file or generated by UFT. The **Parameter** box displays: • The current parameter definition for a value that is already parameterized • The default parameter definition for a value that is not yet parameterized.
📝	**Constant Value Options/Parameter Options.** Opens the Constant Value Options or Parameter Options dialog boxes to edit the value for a constant value, or select a different parameter type or modify the parameter settings for a parameter value. **Note:** When editing a constant value, you can define string or complex value as a regular expression.
Compare image content (for image checkpoints only)	Enables you to compare the expected image source file with the actual image source file. If the expected and actual images are different, UFT displays them both in the Run Results. If the images are identical, only one graphic is displayed.

Current Settings Area (Checkpoint Properties Dialog Box) - Accessibility Checkpoints

Relevant for: GUI tests and scripted GUI components

For an accessibility checkpoint, the Checkpoint Properties dialog box displays a series of possible properties to check in the selected object.



Current settings

- □ ActiveX Check
- ☑ Alt Property Check
- □ Applet Check
- □ Frame Titles Check
- □ Multimedia Links Check
- □ Server-side Image Check
- □ Tables Check

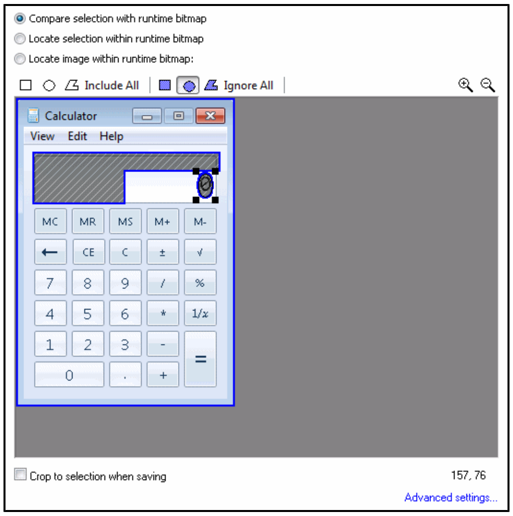
User interface elements are described below:

UI Element	Description
Current settings	(Read-only) The list of the checks to perform. **Note:** You modify the settings for accessibility checkpoints in the Web > Advanced pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Web > Advanced** node). For details, see the section on Accessibility Checkpoint Options in Tests in the *HP Unified Functional Testing Add-ins Guide*.

Bitmap Options Section (Checkpoint Properties Dialog Box) - Bitmap Checkpoints

Relevant for: GUI tests and scripted GUI components

The Bitmap Options area of the Bitmap Checkpoint Properties dialog box enables you to set the area of the bitmap to check as well as recognition criteria for the bitmap images to check.



● Compare selection with runtime bitmap
○ Locate selection within runtime bitmap
○ Locate image within runtime bitmap:

□ Crop to selection when saving

Important Information

- In this area, you define the following:
  - The type of comparison that you want UFT to perform when it runs a bitmap checkpoint step.
    - After selecting a mode, click Advanced settings under the bitmap display area to open the "Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)" on page 1463 and define settings to fine-tune the comparison, or choose a custom comparer to run the comparison.
    - If a custom comparer is selected in the "Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)" on page 1463, you cannot change the type of comparison selected in this area. To enable the selection of a different checkpoint mode. first open the Advanced Settings dialog box and select **UFT Default** from the **Comparer Type** list.
    - **For keyword components only:** The Checkpoint Mode area and the **Advanced settings** option are available only in Advanced Mode.
  - The bitmap that UFT compares with or locates in your application during a run session.

    The options available in this area depend on the checkpoint mode you selected.
    - If you selected **Locate selection within runtime bitmap**, the bitmap to locate is surrounded by a highlighted rectangle. By default, the whole bitmap is selected. You can resize, move, or re-draw the rectangle to specify the area that you want the checkpoint to locate.
    - If you selected **Compare selection with runtime bitmap**, you can manually select areas of the bitmap to include or ignore in the bitmap comparison.

- By default, the whole bitmap is included in the comparison. If you want to select smaller areas to compare, make sure to click **Ignore Entire Bitmap** first, to clear this default.
- Set the toolbar buttons to specify the shape of your selection, and whether to include or ignore it. Then click and drag the cursor to mark the areas you want to select. Finalize a selection by double-clicking in it, or clicking once outside of it. Right-click to cancel a selection before it is finalized.
- You can define multiple areas to compare or ignore.
- If one selection overlaps another, the current selection overrides the previous one, in the overlapping areas. Therefore, for example, if you want to adjust an area that you selected to ignore, you can clear parts of the selection by selecting to include them.
- **For tests and scripted components:** If you define the checkpoint to compare only specific areas of the bitmap, the selected areas are also highlighted in the actual and expected bitmaps displayed in the Run Results Viewer.\
- **For keyword components:** Available only in Advanced Mode.

User interface elements are described below:

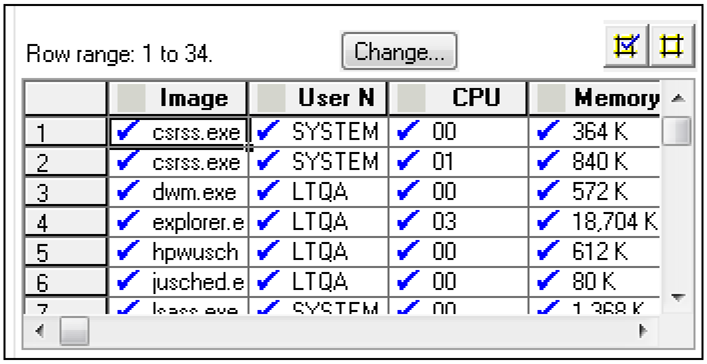
UI Element	Description
Compare selection with runtime bitmap	Instructs UFT to compare the bitmap that you define with the bitmap of the actual object in your application during the run session. • Use the toolbar and editor to select areas to ignore or compare. • Click **Advanced settings** to open the "Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)" on page 1463 and set **Tolerance** settings to fine-tune the comparison.
Locate selection within runtime bitmap	Instructs UFT to check whether the specified bitmap is found anywhere within the bitmap of the actual object in your application during the run session. • In the bitmap editor area (in the Properties grid area of the dialog box), select the area that you want to locate. (By default, the whole bitmap is selected.) • Click **Advanced settings** to open the "Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)" on page 1463 and set **Similarity** settings to fine-tune the comparison.

UI Element	Description
Locate image within runtime bitmap (load image file)	Instructs UFT to check whether the specified image is found anywhere within the bitmap of the actual object in your application during the run session. • Click **Select a bitmap file to load** and browse to a .bmp file containing the image to use for the checkpoint. UFT creates a copy of the selected file and stores it within the checkpoint. For keyword components, the copied file is saved with the checkpoint in ALM. **Note:** If the file you want to load is stored in a network folder, you must map the network folder on your computer before navigating to the file. • Click **Advanced settings** to open the "Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)" on page 1463 and set **Similarity** settings to fine-tune the comparison.
<bitmap selection area>	The area of the selected bitmap to check. Use the toolbar controls to customize the area to check.

Properties Grid Area (Checkpoint Properties Dialog Box) - Database Checkpoints

Relevant for: GUI tests and components

For a database checkpoint, the properties grid area displays a grid with the content of the database object.



Row range: 1 to 34.

	Image	User N	CPU	Memory
1	✓ csrss.exe	✓ SYSTEM	✓ 00	✓ 364 K
2	✓ csrss.exe	✓ SYSTEM	✓ 01	✓ 840 K
3	✓ dwm.exe	✓ LTQA	✓ 00	✓ 572 K
4	✓ explorer.e	✓ LTQA	✓ 03	✓ 18,704 K
5	✓ hpwusch	✓ LTQA	✓ 00	✓ 612 K
6	✓ jusched.e	✓ LTQA	✓ 00	✓ 80 K
7	✓ lsass.exe	✓ SYSTEM	✓ 00	✓ 1,368 K

Important information	- The column header names are captured from the database you selected for your checkpoint. - Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the check are removed from it, and any cells that were not previously included in the check are added to it. - When more than one cell is selected, the options in the Expected Data tab are disabled.

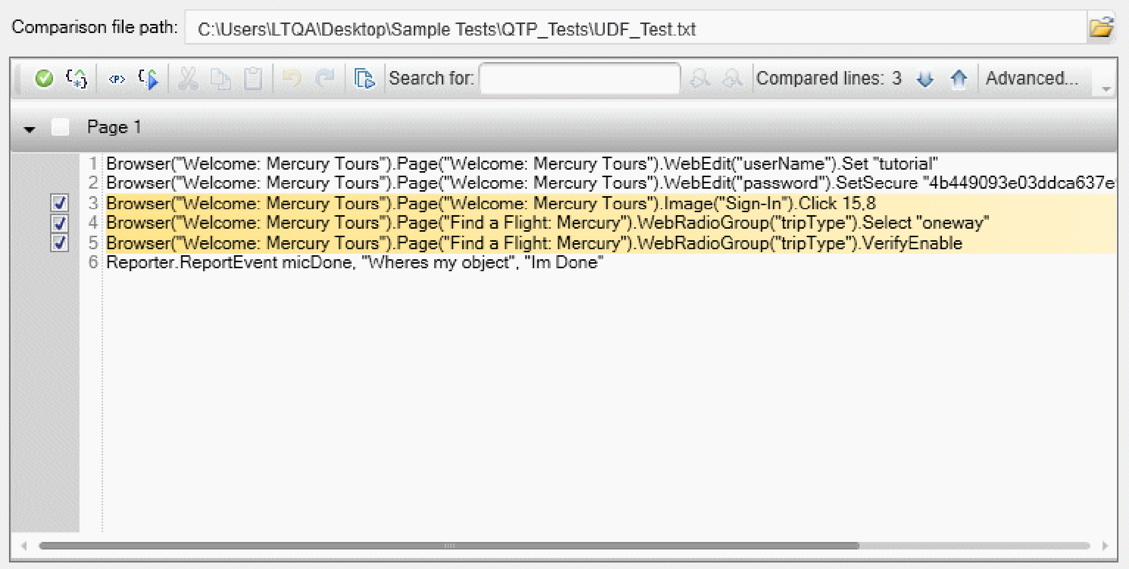
User interface elements are described below:

<grid area>	The grid area displays the captured/expected values of all the cells in the table. Only the ones with blue check marks are used (checked) by the checkpoint. You can check the entire table, specific rows, specific columns, or specific cells.
Change	Modifies the row range. For details, see "Define/Modify Row Range Dialog Box" on page 1467.
☑ ⌗	**Add/Remove from check.** Add or remove the selected cells from the check.

File Content Editor (Checkpoint Properties Dialog Box) - File Content Checkpoints

Relevant for: GUI tests and scripted GUI components

When working with a file content checkpoint, the Properties Grid area displays a File Content Editor which enables you to specify the text to check in a document that is generated or accessed during a run session.



Comparison file path: C:\Users\LTQA\Desktop\Sample Tests\QTP_Tests\UDF_Test.txt

Search for:

Compared lines: 3

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "tutorial"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetSecure "4b449093e03ddca637e
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click 15,8
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebRadioGroup("tripType").Select "oneway"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebRadioGroup("tripType").VerifyEnable
Reporter.ReportEvent micDone, "Wheres my object", "Im Done"
```

User interface elements are described below:

UI Element	Description
Comparison file path	The file path of the generated document that you want to compare during the run session. You can enter a path manually or click the **Select Comparison File** button to navigate to the required file. The path can be relative or absolute (unless you use a regular expression). The file must be located in the file system. When you initially specify a file to compare, the textual content from that file is displayed in the file content editor of the dialog box as the expected value of the checkpoint. If you later change this path, the content in the file content editor is not affected. **Note:** The document must not be password-protected, otherwise, UFT will not be able to access it during a run session. You can use the following when specifying the document path: • **Configure Verification Value (Parameter/Regular Expression).** Opens the "Value Configuration Options Dialog Box" (described on page1579). You use this dialog box to define the file path as a constant or parameter value, with or without a regular expression. For example, suppose your application is being released in multiple languages, and you want to compare a different source file for each language. You can specify a data table parameter to instruct UFT to use a different file for each iteration. Displayed only when you hover over the **Comparison file path** text box. **Note:** If you specify a parameter value for the comparison file path and multiple files that match the value exist in the specified location, UFT performs the checkpoint on the latest generated file (according to its creation time). If you use a regular expression, you must specify a mapped drive, for example: C:\\Program Files\\HP\\Unified Functional Testing\\Tests\\.*\.txt • **Select Comparison File.** Enables you to navigate to the file you want to compare.

UI Element	Description
✓	**Select/Clear Line for Verification.** This toggle button enables you to select or clear a line to compare with the generated file during a run session. When you select a line, UFT highlights it and selects the check box to the left of the line. (Also available from the context menu)
{ }	**Treat Line as Regular Expression/Plain Text.** This toggle button enables you to instruct UFT to look for and handle regular expressions in the line. **Note:** The line does not need to be selected for comparison to insert a regular expression (or to modify the text in general). (Also available from the context menu)
<P>	**Add Parameter to Line.** Opens the "Value Configuration Options Dialog Box" (described on page 1579). You use this dialog box to define a parameter as a constant or parameter value. For example, suppose you inserted a parameter in the "Comparison file path" box because you want to compare a different file during each iteration. You can specify a data table parameter to instruct UFT to use a different value for each iteration. (Also available from the context menu)
{ ▶	**Open Regular Expression Evaluator.** Opens the "Value Configuration Options Dialog Box" (described on page 1579). You use this dialog box to create and test a regular expression enabling you to determine whether it suits your needs. (Also available from the context menu)

UI Element	Description
	Preview Comparison. Opens a comparison window showing the differences between the source file in the file system and the currently displayed file in the file content editor. Usage examples: • Suppose you modified the text in the file content editor by adding parameters and deleting some text and lines, you may want to compare the current content to the text in the original file to make sure that your checkpoint is still applicable. • Suppose the original file in the file system was modified. You can compare the content in that file with the content of the text in the file content editor. **Tip:** If the text in the source file is very different from the text in the file content editor, you may want to recreate the checkpoint. **Note:** • If you select **Checkpoint Properties** in the "Advanced File Content Checkpoint Properties Dialog Box"(described on page 1461), these properties are ignored during a preview comparison. (**Comparison Settings**, however, are included in the preview comparison.) • This view is similar to the comparison available in the Captured Data pane in the Run Results Viewer for failed file content checkpoint steps. For details, see the section on File Content Checkpoint Results (described in the *HP Run Results Viewer User Guide*).
--	**Search for.** Text box in which you can optionally enter text for which to search. As you type, all instances of the specified text are highlighted. Press ENTER to jump to the first instance if it is not visible in the file content editor. You can use the following buttons to navigate between each instance of the specified text: **Find Next.** Jumps to the next instance of the specified text in the file content editor. **Find Previous.** Jumps to the previous instance of the specified text in the file content editor. (Enabled only after you click **Find Next**.)
Advanced	Opens the "Advanced File Content Checkpoint Properties Dialog Box". This dialog box enables you to set additional comparison settings and checkpoint properties.

UI Element	Description
Page	The page from the source file being compared. Pages are displayed only when the content from the source file exceeds one page. You can: • View the number of pages in the file being compared. • Expand or contract a specific page to view or hide the content on that page. • Select the adjacent check box to select all of the lines on a page. • Clear the adjacent check box to clear the selection of all of the lines on a particular page.

Properties Grid Area (Checkpoint Properties Dialog Box) - Table Checkpoints

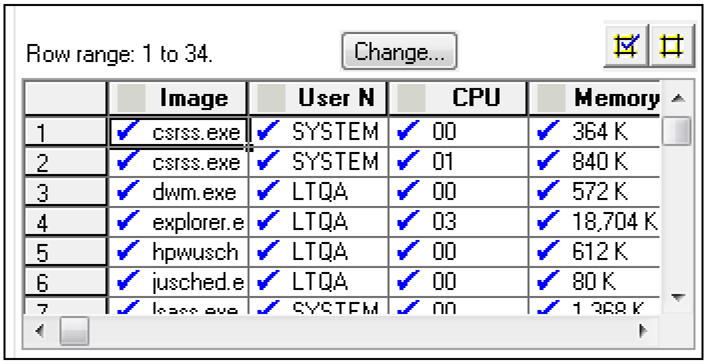
Relevant for: GUI tests and scripted GUI components

For a table checkpoint, the Properties Grid area displays two tabs:

- "Table Content Tab" below
- "Properties tab" on the next page

Table Content Tab

In the Table Content, the dialog box displays a grid with the content of the selected Table object.



Row range: 1 to 34.

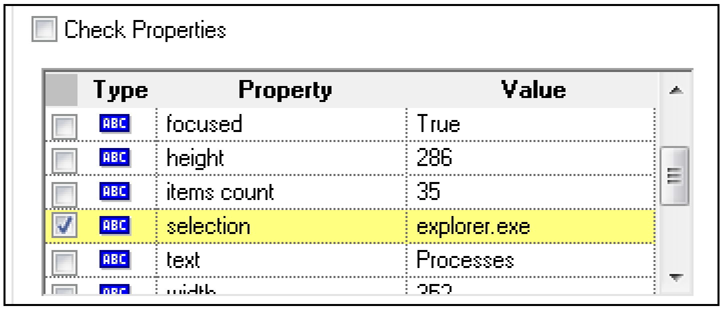
	Image	User N	CPU	Memory
1	✓ csrss.exe	✓ SYSTEM	✓ 00	✓ 364 K
2	✓ csrss.exe	✓ SYSTEM	✓ 01	✓ 840 K
3	✓ dwm.exe	✓ LTQA	✓ 00	✓ 572 K
4	✓ explorer.e	✓ LTQA	✓ 03	✓ 18,704 K
5	✓ hpwusch	✓ LTQA	✓ 00	✓ 612 K
6	✓ jusched.e	✓ LTQA	✓ 00	✓ 80 K
7	✓ lsass.exe	✓ SYSTEM	✓ 00	✓ 1,368 K

User interface elements are described below:

<grid area>	The grid area displays the captured/expected values of all the cells in the table. Only the ones with blue check marks are used (checked) by the checkpoint. You can instruct UFT to check the entire table, specific rows, specific columns, or specific cells. **Note:** UFTchecks only cells containing a check mark.
Change	Enables you to define or modify the row range to check by opening the "Define/Modify Row Range Dialog Box" on page 1467.
☑ ⌗	**Add/Remove from check.** Add or remove the selected cells from the check.

Properties tab

In the Properties tab, the dialog box displays the properties of the Table object (without the table content).



□ Check Properties

	Type	Property	Value
□	ABC	focused	True
□	ABC	height	286
□	ABC	items count	35
☑	ABC	selection	explorer.exe
□	ABC	text	Processes
□	ABC	width	252

User interface elements are described below:

UI Elements	Description
Check box	For each object class, UFT recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.

UI Elements	Description
Type	• ABC . Indicates that the value of the property is currently a constant. • . Indicates that the value of the property is currently a test or action parameter (tests only). • . Indicates that the value of the property is currently a Data Table parameter. (tests and scripted components only) • . Indicates that the value of the property is currently an environment variable parameter. (tests and scripted components only) • . Indicates that the value of the property is currently a random number parameter. (tests and scripted components only) • . Indicates that the value of the property is currently a parameter. (keyword components only)
Property	The name of the property.
Value	The expected value of the property. For details on modifying property values, see "Configure Value Area" on page 1434. **For keyword components:** See also: "Parameterization / Properties Dialog Box (Checkpoints)" on page 1473.

Configure Value Area (Checkpoint Properties Dialog Box) - Table/Database Checkpoints

Relevant for: GUI tests and scripted GUI components

For a table checkpoint (in the Table Content Tab) and a database checkpoint, this area displays three tabs:

- "Expected Data Tab" on the next page
- "Settings Tab" on the next page
- "Cell Identification Tab" on page 1450

> **Note:** For a table checkpoint, the Properties tab also displays the Configure Value area, For details, see "Configure Value Area (Checkpoint Properties Dialog Box) - Standard/Image Checkpoints" on page 1436.

Expected Data Tab

Important information	When more than one cell is selected (highlighted) in the grid area, the options in the Expected Data tab are disabled.

User interface elements are described below:

UI Elements	Description
Selected cell	Indicates the table name and the row and column numbers of the selected cell.
Configure value	Enables you to set the expected value of the cell as a constant or parameter. For details on modifying values, see "Configure Value Area (Checkpoint Properties Dialog Box) - Standard/Image Checkpoints" on page 1436.
✎	**Constant Value Options/Parameter Options.** Opens the Constant Value Options or Parameter Options dialog boxes to edit the value for a constant value, or select a different parameter type or modify the parameter settings for a parameter value. **Note:** When editing a constant value, you can define string or complex value as a regular expression.

Settings Tab

Important information	• The settings in this tab apply to all cells marked for verification. • The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.

User interface elements are described below:

Option	Description
Verification type	Specifies how cell contents are compared: • **String Content.** (Default) Evaluates the content of the cell as a string. For example, 2 and 2.00 are not recognized as the same string. • **Numeric Content.** Evaluates the content of the cell according to numeric values. For example, 2 and 2.00 are recognized as the same number. • **Numeric Range.** Compares the content of the cell against a numeric range, where the minimum and maximum values are any real number that you specify. This comparison differs from string and numeric content verification in that the table data is compared against the range that you defined and not against a specific expected value.

Option	Description
Exact match	(Default) Checks that the exact text, and no other text, is displayed in the cell. Clear this check box if you want to check that a value is displayed in a cell as part of the contents of the cell. Available only when **String Content** is selected as the **Verification type**.
Ignore space	(Default) Ignores spaces in the captured content when performing the check. The presence or absence of spaces does not affect the outcome of the check. Available only when **String Content** is selected as the **Verification type**.
Match case	Conducts a case sensitive search. Available only when **String Content** is selected as the **Verification type**.
Min / Max	Specifies the numeric range against which the content of the cell is compared. The range values can be any real number. Available only when **Numeric Range** is selected as the **Verification type**.

Cell Identification Tab

Important information	The settings in this tab apply to all cells marked for verification.

User interface elements are described below:

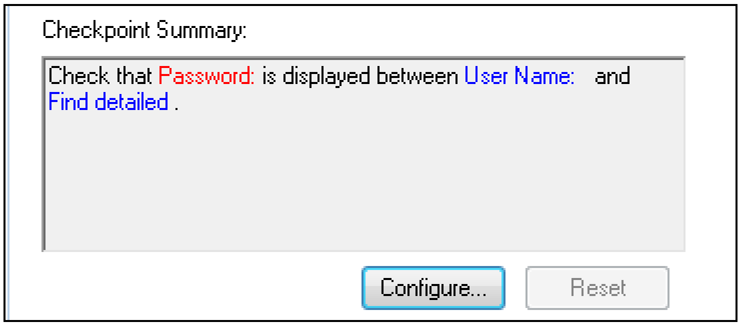
Identify columns	Specifies the location of the column (in your actual table) containing the cell(s) to which you want to compare the expected data. • **By position.** (Default) Locates cells according to the column position. A shift in the position of the columns within the table results in a mismatch. • **By column name.** Locates cells according to the column name. A shift in the position of the columns within the table does not result in a mismatch. (Enabled only when the table contains more than one column.)

Identify rows	Specifies the location of the row (in your actual table) containing the cell(s) to which you want to compare the expected data. - **By row number.** (Default) Locates cells according to the row position. A shift in the position of any of the rows within the table results in a mismatch. - **By selected key column(s).** Locates the row(s) containing the cells to be checked by matching the value of the cell whose column was previously selected as a **key column**. A shift in the position of the row(s) does not result in a mismatch. If more than one row is identified, UFT checks the first matching row. You can use more than one key column to uniquely identify any row. **Note:** A key symbol 🔑 is displayed in the header of selected key columns.
Use value match criteria to identify data in the key column	Instructs UFT to use the verification type settings from the Settings tab as the criteria for identifying data in the key column. Enabled only when you select to identify rows **By selected key column(s)**.

Checkpoint Summary Area (Checkpoint Properties Dialog Box) - Text/Text Area Checkpoints

Relevant for: GUI tests and components

For a text or text area checkpoint, the Properties Grid area displays a summary of the text to check.



Checkpoint Summary:

Check that Password: is displayed between User Name: and Find detailed .

User interface elements are described below:

UI Element	Description
Checkpoint Summary	Summarizes the selected text for the checkpoint. It displays the text you selected when creating the checkpoint, plus the text before and after it. UFT automatically displays the checked text in red, and the text before and after the checked text in blue. For text checkpoints in Web-based environments, it displays the text you selected when creating the checkpoint, plus some text before and after it. For text and text area checkpoints in Windows-based environments, it displays the text you selected when creating the checkpoint. **Note:** In Windows-based environments, if there is more than one line of text selected, the Checkpoint Summary area displays **[complex value]** instead of the selected text string. You can click **Configure** to view and manipulate the actual selected text for the checkpoint.
Configure	Opens the "Configure Text Selection Dialog Box" (described on page 1466), where you can specify the checked text, the text before (if any), and the text after (if any).
Reset	Resets the text selection to the previous configuration.

Checkpoint Options Area (Checkpoint Properties Dialog Box) - Text/Text Area Checkpoints

Relevant for: GUI tests and scripted GUI components

When working with a Text or Text Area checkpoint, the Configure Value area displays three different sets of options, depending on if you are editing the text to check, the text before the checked text, or the text after the checked text:

- "Checked Text Option" on the next page
- "Text Before Option" on the next page
- "Text After Option" on page 1454

Checked Text Option

User interface elements are described below:

UI Element	Description
Constant	(Default) Sets the expected value of the checked text as a constant. For details on modifying values, see "Configure Value Area " on page 1575. **Tip:** The **Constant** box displays the checked text. You can change the checked text by typing in the **Constant** box or by using the "Configure Text Selection Dialog Box" (described on page 1466).
Parameter	Sets the expected value of the checked text as a parameter. For details on modifying values, see "Configure Value Area " on page 1575.
Match case	Conducts a case-sensitive check.
Exact match	Checks for the exact expected text. For example, if you create a checkpoint with the following description, Check that New York is displayed between Flight departing from and to San Francisco, and select **Exact match**, if the actual text is New York City, the checkpoint fails. If you do not select **Exact match**, the checkpoint passes because the expected text is contained within the actual text.
Ignore spaces	Ignores spaces in the captured text when performing the check. The presence or absence of spaces does not affect the outcome of the check.
Text not displayed	Checks that the text string is not displayed. For example, if you create a checkpoint with the following description, Check that New York is displayed between Flight departing from and to San Francisco, and select **Text not displayed**, UFT checks that the text **New York** is not displayed.

Text Before Option

User interface elements are described below:

UI Element	Description
Use the text before	Checks the text before the checked text. To ignore this text, clear this check box.

UI Element	Description
Text to check is displayed after occurrence _ of	Checks that the checked text is displayed after the specified text. If the identical text string you specify is displayed more than once on the page, you can specify the occurrence of the string to which you are referring. If you accept the default text that UFT recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is accurate. If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words Mercury Tours are displayed after the fourth occurrence of the word the, enter 4 in the **Text to check is displayed after occurrence** box.
Constant	(Default) Sets the expected value of the text before the checked text as a constant. For details on modifying values, see "Configure Value Area " on page 1575. If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1. **Tip:** The **Constant** box displays the text before the checked text. You can change the text by typing in the **Constant** box or by using the "Configure Text Selection Dialog Box" on page 1466.
Parameter	Sets the expected value of the text before the checked text as a parameter. For details on modifying values, see "Configure Value Area " on page 1575.

Text After Option

User interface elements are described below:

UI Element	Description
Use the text after	Checks the text after the checked text. To ignore this text, clear this check box.

UI Element	Description
Text to check is displayed before occurrence _ of	Checks that the checked text is displayed before the specified text. If the identical text string you specify is displayed more than once on the page, you can specify to which occurrence of the string you are referring. **Note:** UFT starts counting occurrences of the specified **Text After** value from the beginning of the text string you selected to output, and includes any occurrences within the output value string itself. If you accept the default text that UFT recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate. If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words Mercury Tours are displayed before the fourth occurrence of the word the (where occurrences are counted only at the end of the text defined in the **Text Before** option area), enter 4 in the **Text to check is displayed before occurrence** box.
Constant	(Default) Sets the expected value of the text after the checked text as a constant. For details on modifying values, see "Configure Value Area " on page 1575. If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1. **Tip:** The **Constant** box displays the text after the checked text. You can change the text by typing in the **Constant** box or by using the "Configure Text Selection Dialog Box" on page 1466.
Parameter	Sets the expected value of the text after the checked text as a parameter. For details on modifying values, see "Configure Value Area " on page 1575.

Configure Value Area (Checkpoint Properties Dialog Box) - Page Checkpoints

Relevant for: GUI tests and scripted GUI components

When working with a Page checkpoint, the Configure Value area displays three separate sections:

- "Configure Value Area" below
- "HTML Verification Area" on the next page
- "All Objects in Page Area" on the next page

Configure Value Area

For details on the user interface elements in this area, see "Configure Value Area " on page 1575.

HTML Verification Area

Important: You can select these options only when creating a Page checkpoint while recording.

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

UI Element	Description
HTML source	Checks that the source in the Web page being tested matches the expected HTML code (the source code of the page at the time that the test or scripted component is recorded). Available only when creating a page checkpoint while recording.
Edit HTML Source (enabled only when the HTML Source check box is selected)	Opens the HTML Source dialog box, which displays the expected HTML code. Edit the expected HTML source code and click **OK**. Note that you can also use regular expressions when editing the expected HTML source code if you click the regular expression check box at the bottom of the page.
HTML tags	Checks that the HTML tags in the Web page being tested match the expected HTML tags (the HTML tags on the page at the time that the test or scripted component is recorded).
Edit HTML Tags (enabled only when the HTML Tags check box is selected)	Opens the dialog box that displays the expected HTML tags. Edit the expected HTML tags and click **OK**. Note that you can also use regular expressions when editing the HTML tags if you click the regular expression check box at the bottom of the page.

All Objects in Page Area

User interface elements are described below:

UI Element	Description
Links	Checks the functionality of the links in the page according to your selections in the "Filter Link Check Dialog Box".
Filter Link Check (enabled only when the Links check box is selected)	Opens the "Filter Link Check Dialog Box", which enables you to specify which hypertext links to check in the page.
Images	Checks that the images are displayed on the page according to your selections in the "Filter Link Check Dialog Box".

UI Element	Description
Filter Image Check (enabled only when the Images check box is selected)	Opens the "Filter Link Check Dialog Box" (described on page 1470), which enables you to specify which image sources to check in the page.
Broken links	Instructs UFT to check for broken links. **Note:** If you want to check only links that are targeted to your current host, select the **Broken links - checks only links to current host** option in the Web pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Web > General** node. For details, see the section on setting Web testing options in the *HP Unified Functional Testing Add-ins Guide*.

XML Tree/Options Area (Checkpoint Properties Dialog Box) - XML Checkpoints

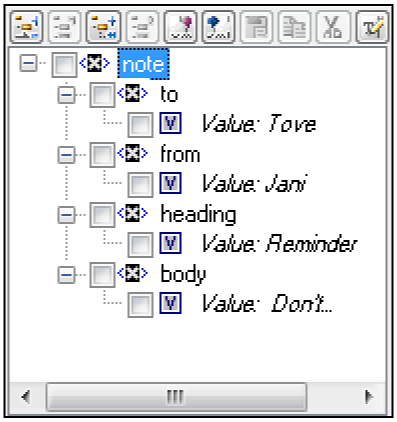
Relevant for: GUI tests and scripted GUI components

For an XML checkpoint, this area displays the XML hierarchy of the selected test object and the elements, attributes, and values to check.

This area includes the following sections:

- "XML Tree" below
- "Options Area" on page 1459

XML Tree

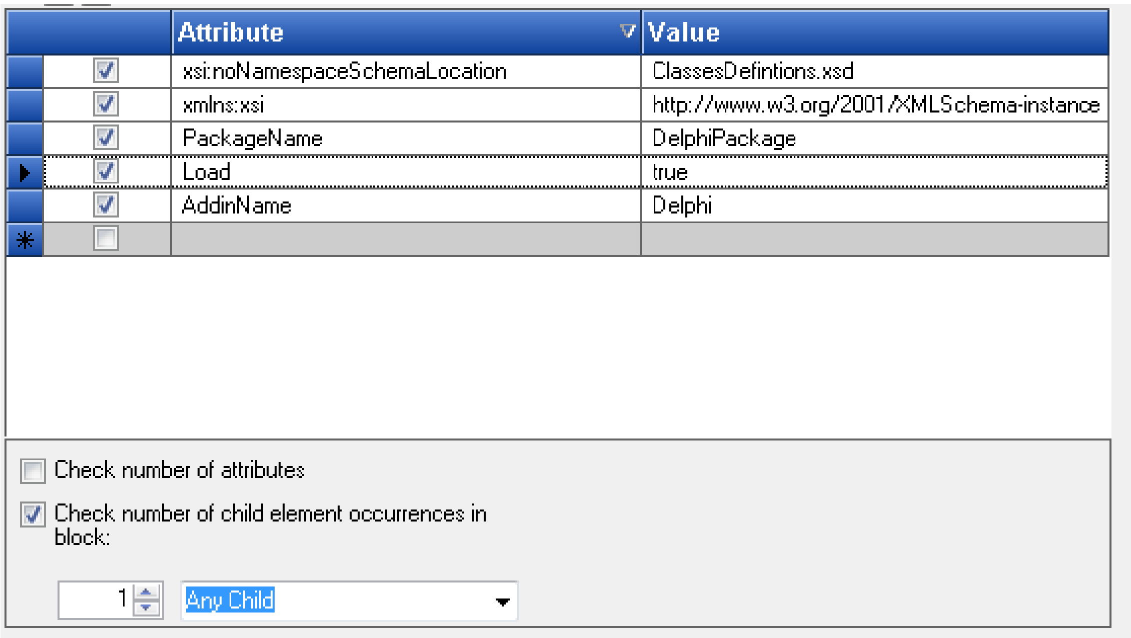


Important information	- The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the specific elements, attributes and values you want to check. Each element is displayed with a <⊠> icon. Each value is displayed with a V icon. - Select the check box next to an element or value node to include that item in the checkpoint. Select an element or value node in the XML tree to display, edit, or parameterize its expected attributes and values in the options area of the XML Checkpoint Properties dialog box. For details, see the section on the Options area, below.

User interface elements are described below:

UI Element	Description
	Add Child. Adds a child node below the selected node in the tree.
	Insert Sibling. Adds a sibling node at the same level as the selected node in the tree.
	Add Value. Enables you to assign a constant or parameterized value to the selected element.
	Delete. Deletes the selected node. Note that you cannot delete the root node of the checkpoint.
	Import XML. Enables you to browse to an existing XML file and import it. The new file replaces the selected node and its current sub-tree.
	Export XML. Enables you to save the content of the checkpoint tree to an .xml file. This option is enabled only when the root node of the tree is selected.
	Edit XML as Text. Opens the "Edit XML as Text Dialog Box" (described on page 1469), enabling you to modify the XML text of the selected node and its subnodes in a text editor.
	Select All. Selects all element and value nodes in the XML tree as well as all element attributes.
	Clear All. Clears all element and value nodes in the XML tree as well as all element attributes.
	Duplicate. Adds a new node, identical to the selected one, as a sibling node at the same level as the selected node in the XML tree. **Available:** Only from the context menu (right-click menu).
<XML tree>	A hierarchy of all XML elements in the selected XML test object.

Options Area



		Attribute	Value
	☑	xsi:noNamespaceSchemaLocation	ClassesDefintions.xsd
	☑	xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
	☑	PackageName	DelphiPackage
▶	☑	Load	true
	☑	AddinName	Delphi
*	□		

□ Check number of attributes

☑ Check number of child element occurrences in block:

1 Any Child

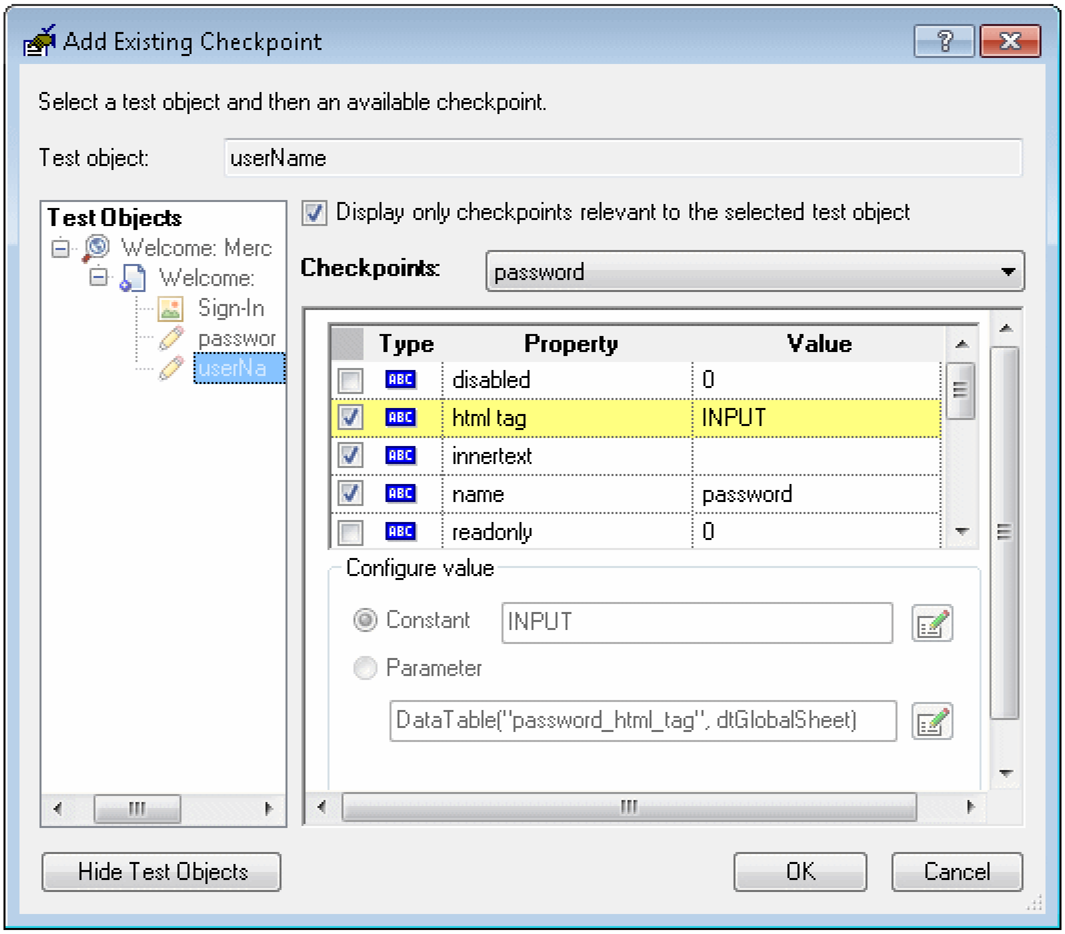
User interface elements are described below:

UI Element	Description
Attribute	The list of attributes for the selected element or value node in the XML tree.
Value	The list of values for the selected element or value node in the XML tree.
Check number of attributes	Checks the number of attributes that are attached to the element.
Check number of child element occurrences in block	Displays the number of child elements associated with the selected parent element. If you select this option, UFT verifies that the number of child elements in your XML tree (with the specified name, if applicable) corresponds to the number that appears in the **Check number of child element occurrences in block** field. You can specify the child element name for the Number of child element occurrences check. If you select a child element name, UFT verifies that the number of child elements with that name corresponds to the number that you specify in the **Number of child element occurrences in block** field. Select **Any Child** (default) to check the total number of child elements associated with the selected parent element.
Schema Validation	Confirms that the XML in your application or file adheres to the structure defined in a specific XML schema or schemas. You can validate the structure of the XML you are checking using one or more external schema files or using schemas embedded within your XML document. For details, see "Schema Validation Dialog Box" on page 1476.

Add Existing Checkpoint Dialog Box

Relevant for: GUI tests and scripted GUI components only

This dialog box enables you to add an existing checkpoint to your test while editing.



Select a test object and then an available checkpoint.

Test object: userName

☑ Display only checkpoints relevant to the selected test object

Checkpoints: password

	Type	Property	Value
□	ABC	disabled	0
☑	ABC	html tag	INPUT
☑	ABC	innertext	
☑	ABC	name	password
□	ABC	readonly	0

Configure value

● Constant: INPUT

○ Parameter: DataTable("password_html_tag", dtGlobalSheet)

To access	1. Ensure that a GUI action or component is the active document in the document pane. 2. Select **Design > Checkpoint > Existing Checkpoint.**
Important information	• This option is available only if at least one of the object repositories associated with the current action (including the local object repository) contains at least one checkpoint. • If a test object step is highlighted in the Keyword View or the cursor is located in a step in the Editor, the Add Existing Checkpoint dialog box opens with the **TestObjects** tree hidden. The test object displayed in the **Test object** box is the object from the highlighted step in the Keyword View or the specific object where the cursor is located in the Editor.
See also	"Adding Existing Checkpoints to a Test" on page 1396

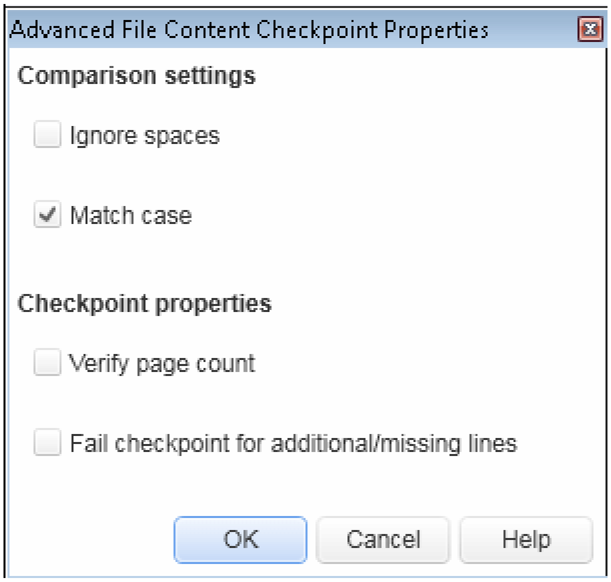
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Test object	The name of the test object for which you are adding a checkpoint.
Test Objects tree	All objects in the current action.
Show/Hide Test Objects	Shows or hides the **Test Objects** tree.
Display only checkpoints relevant to the selected test object	When selected, UFT determines which checkpoints from the current action's object repositories are relevant for the selected object (based on the checkpoint type and the properties selected in the checkpoint) and displays only those checkpoints in the **Checkpoints** list. When using this option, open your application and display the selected object to enable UFT to accurately determine all of the checkpoints that can apply to that object.
Checkpoints	Lists the checkpoints available for insertion. If the **Display only checkpoints relevant to the selected test object** option is cleared, this list includes all checkpoints from all object repositories associated with the current action. If the **Display only checkpoints relevant to the selected test object** option is selected, this list displays only the relevant checkpoints as described above.
<checkpoint details area>	Displays the settings of the selected checkpoint in read-only format.
Configure value	The value for the selected checkpoint in read-only mode. For details, see "Value Configuration and Parameterization" on page 1570.

Advanced File Content Checkpoint Properties Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to configure how the checkpoint is performed.



Advanced File Content Checkpoint Properties

Comparison settings

□ Ignore spaces

☑ Match case

Checkpoint properties

□ Verify page count

□ Fail checkpoint for additional/missing lines

To access	In the "Checkpoint Properties Dialog Box" (described on page 1432), select **Advanced** in the toolbar.
Important information	When a file content checkpoint step runs, it locates the checkpoint text within the line by comparing all of the text on that line. The options in this dialog box instruct UFT how to treat the text on every line in the file in which one or more checkpoints are specified for a single checkpoint object. (You can create multiple checkpoints in a single file—all within a single checkpoint object.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
See also	"Checkpoint Properties Dialog Box" on page 1432

User interface elements are described below:

UI Element	Description
Ignore spaces	Ignores extraneous spaces on the same line as the checked text when comparing the expected content with the actual content. This is useful when there might potentially be one or more spaces in the text preceding or following the checked text, and you do not want the checkpoint to fail because of this. **Note:** This setting does not affect **missing** spaces, which are treated as non-matching text. **Default:** Cleared

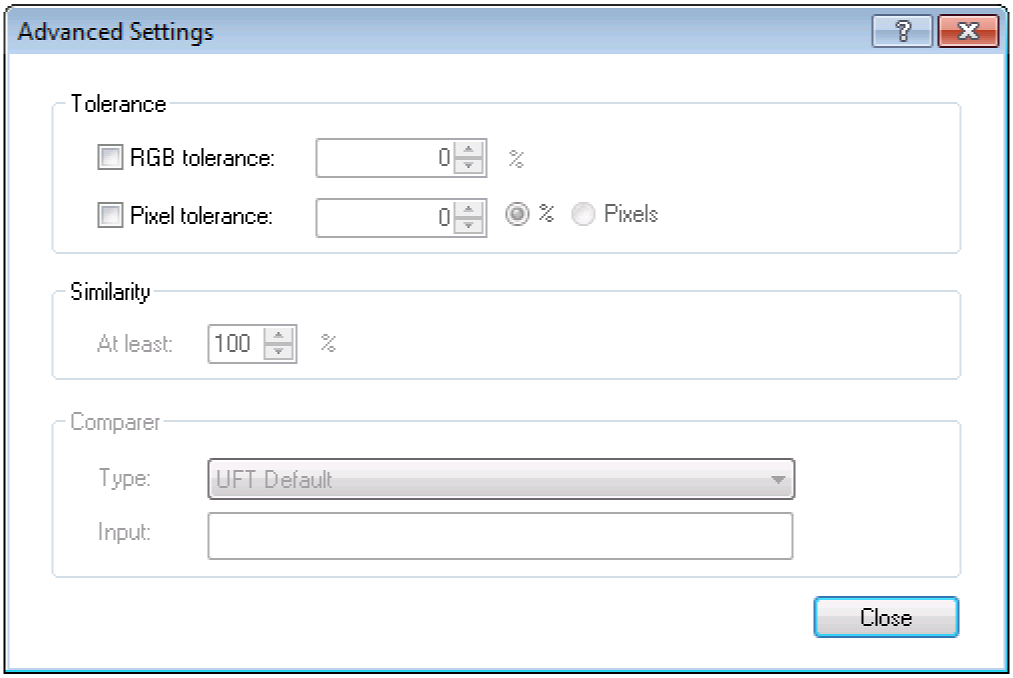
UI Element	Description
Match case	Conducts a case-sensitive check of all the text on the same line as the checked text when comparing the expected content with the actual content. **Default:** Selected
Verify page count	Compares the number of pages in the checkpoint object with the number of pages in the file generated or accessed during the run session. When this option is selected, the checkpoint fails if the page count comparison is not identical. **Note:** This setting is ignored during a preview comparison. **Default:** Cleared
Fail checkpoint for additional/missing lines	Compares the number of lines in the checkpoint object with the number of lines in the file generated or accessed during the run session. When this option is selected, the checkpoint fails if one or more lines was added or removed in the file being compared. **Note:** This setting is ignored during a preview comparison. **Default:** Cleared

Advanced Settings Dialog Box (Bitmap Checkpoints Dialog Box)

Relevant for: GUI tests and components

This dialog box enables you to define advanced settings for a bitmap checkpoint, in order to fine-tune the comparison that UFT performs when running the checkpoint.

The following image shows an example of the Advanced Settings dialog box when the **Compare selection with runtime bitmap** option is selected in the Bitmap Checkpoint Properties dialog box, and no custom comparers are installed and registered on the computer.



Advanced Settings

Tolerance

□ RGB tolerance: 0 %

□ Pixel tolerance: 0 ● % ○ Pixels

Similarity

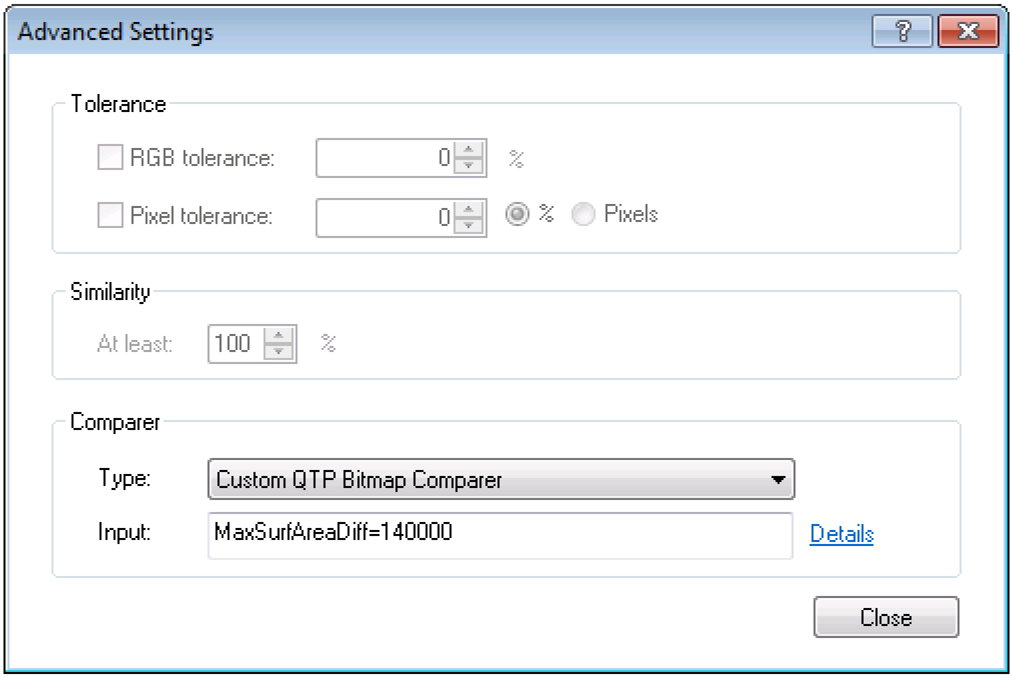
At least: 100 %

Comparer

Type: UFT Default

Input:

The following image shows an example of the Advanced Settings dialog box when a custom comparer is selected in the Comparer **Type** list. (The **Comparer** options are enabled because a custom comparer is installed and registered on the UFT computer, and



Advanced Settings

Tolerance

□ RGB tolerance: 0 %

□ Pixel tolerance: 0 ● % ○ Pixels

Similarity

At least: 100 %

Comparer

Type: Custom QTP Bitmap Comparer

Input: MaxSurfAreaDiff=140000

To access	In the "Checkpoint Properties Dialog Box" (described on page 1432), click **Advanced settings** beneath the bitmap display area (available for keyword components only in Advanced Mode).

Important information	The options enabled in this dialog box differ, depending on whether you are comparing an expected bitmap with a runtime bitmap or locating a specified bitmap within the runtime bitmap.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
See also	• "Checkpoint Properties Dialog Box" on page 1432 • "Fine-Tuning the Bitmap Comparison" on page 1403

User interface elements are described below:

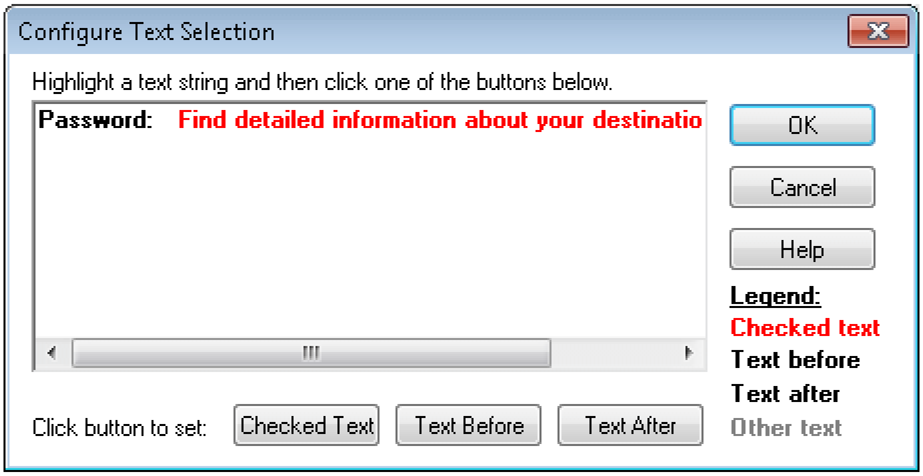
UI Element	Description
Tolerance Relevant only when the **Compare selection with runtime bitmap** option is selected.	
RGB tolerance	Enables you to define the percent by which the RGB values of the pixels in the runtime bitmap can differ from those of the expected bitmap and allow the checkpoint to pass. Available only for bitmaps with a color depth of 24 bits.
Pixel tolerance	Enables you to define the number or percentage of pixels in the runtime bitmap that can differ from those in the expected bitmap and allow the checkpoint to pass. Select either the **Percent** or **Pixels** radio button, and set the relevant value. If you switch between the **Percent** and **Pixels** radio buttons after entering the tolerance value, the value is recalculated based on your selection. (100% is the total number of pixels in the expected bitmap or selected area.)
Similarity Relevant only when the **Locate selection within runtime bitmap** or **Locate image within runtime bitmap (load image file)** option is selected.	
At least	Enables you to determine the amount of differences in pixels between the specified bitmap that you are locating and the runtime bitmap that can occur and still allow the checkpoint to pass. If used, then UFT may locate possible candidates for the specified bitmap that you are attempting to locate, and display the possible candidates in the Run Results Viewer, in addition to the results of the checkpoint.
Comparer Relevant only: • When the **Compare selection with runtime bitmap** option is selected. • If one or more custom comparers are installed and registered on the UFT computer. For details, see "Fine-Tuning the Bitmap Comparison" on page 1403.	

UI Element	Description
Type	Enables you to select the comparer for UFT to use to run the checkpoint. You can select the UFT default comparer or a custom comparer. A custom comparer is a COM object that you or a third party develop to run the bitmap comparison in the checkpoint according to a more specific algorithm.
Input	Enables you to provide input (in string format) to the custom comparer, for any configuration options it supports. By default, this box displays a configuration string provided by the custom comparer (if available). **Example:** You might be able to specify tolerance levels, an acceptable deviation in size or location of the bitmap, and so on. Relevant only if a custom comparer is selected from the Comparer **Type** list.
Details	Opens help information provided by the custom comparer (if available). This help can include instructions for providing configuration input to the comparer, information about the algorithm that the custom comparer uses to compare the bitmaps, an explanation about when to use this custom comparer, and so on. Available only if a custom comparer is selected from the Comparer **Type** list, and documentation is provided for this comparer.

Configure Text Selection Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to specify the checked text, the text before, and the text after, for a text checkpoint.


Configure Text Selection

Highlight a text string and then click one of the buttons below.

Password: Find detailed information about your destinatio

To access	In the Checkpoint Summary Area of the Checkpoint Properties dialog box, click the **Configure** button in the **Checkpoint Summary** area.
Important information	• UFT displays the checked text in red and the text before and after it in black (as indicated in the **Legend** displayed in the dialog box). • To remove text from the current text selection configuration, highlight only the text you want included as the before or after text and click the appropriate button. Any text that is not selected as **Checked Text**, **Text Before**, or **Text After** is displayed in gray. The gray text is not displayed the next time the Configure Text Selection dialog box is opened. **Example:** In the sample image above, suppose you wanted to check only the word information, and you wanted UFT to look for this text between Find detailed and about your destination, then: • Highlight the word information, and click **Checked Text**. The word information remains red, and the other text turns black. • Highlight the words Find detailed and click **Text Before**. The words Find detailed remain black, and all text preceding it turns gray. This gray text will be removed from the text configuration when you click **OK**. • The words about your destination are already marked in black as the text after, so there is no need to modify this configuration.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419

To modify which text is checked and how that text is found, based on the text before and after it, highlight the text you want to set for one of these items and then click the appropriate button:

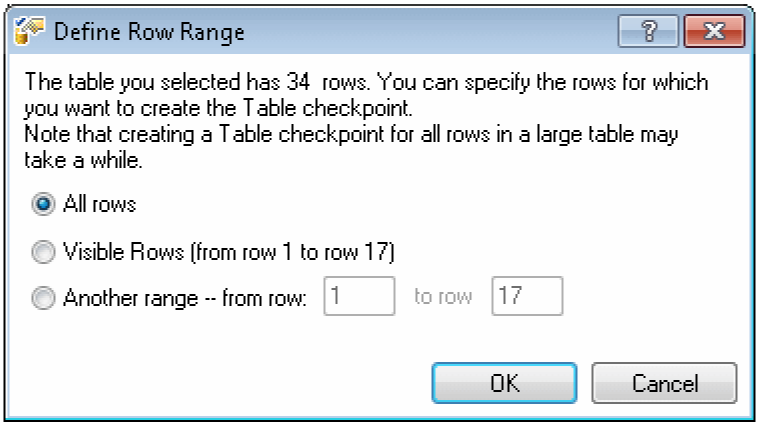
UI Elements	Description
Checked Text	Sets the highlighted text as the checked text. UFT displays this text in red and the remainder in black.
Text Before	Sets the highlighted text as the text before the checked text.
Text After	Sets the highlighted text as the text after the checked text.

Define/Modify Row Range Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to define or modify the number of rows included in an existing table or database checkpoint.

The following image shows this dialog box as it appears for a table checkpoint in define mode. Depending on the mode of the dialog box, the actual title bar of the image may differ. This dialog box may also differ when viewed for a table output value object.



The table you selected has 34 rows. You can specify the rows for which you want to create the Table checkpoint.
Note that creating a Table checkpoint for all rows in a large table may take a while.

● All rows

○ Visible Rows (from row 1 to row 17)

○ Another range -- from row: 1 to row 17

To access	In the Table Checkpoint Properties dialog box or Table Output Value Properties dialog box, click the **Change** button.
Important information	This dialog box is also available for certain list view objects. This enables you to define a table checkpoint on specific rows within a list view object.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
See also	• "Checkpoint Properties Dialog Box" on page 1432 • "Considerations for Modifying Row Ranges" on the next page • "How to Create or Modify an Output Value Step" on page 1494

User interface elements are described below:

UI Elements	Description
All rows	Includes all rows in the checkpoint or output value. **Note:** Capturing all of the data for large table or list view objects may take some time.
Visible Rows	Includes only the rows that are currently visible in your application in the checkpoint or output value. Available only for some environments or object types.
Another range	Includes the rows you specify in the checkpoint or output value. You can specify any row range between 1 and the number of rows listed in the dialog box.

Considerations for Modifying Row Ranges

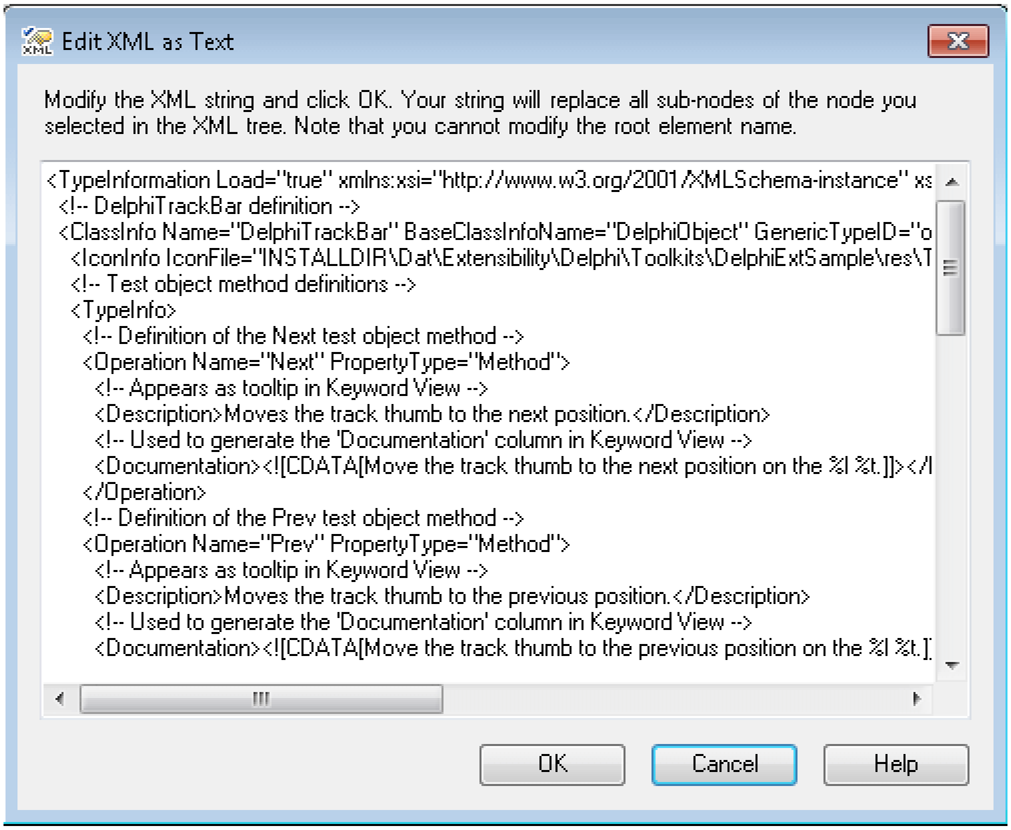
- If your modified row range includes new rows, UFT captures the current values of the new rows from the open application.
- If your modified row range includes some or all of the rows that were already included in the checkpoint, the expected values of those cells are not changed. This enables you to modify the row range without losing parameterization, regular expressions, or other changes you may have made to the expected cell values in your checkpoint.

  Therefore, you cannot use the Modify Row Range dialog box to update the expected values of an existing table checkpoint. To update the expected values of your checkpoint, use the **Update Run Mode** option. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.
- If your modified row range excludes some or all of the rows that were previously included in your checkpoint, those rows (and any modifications you made to the expected values) are deleted from the checkpoint.

Edit XML as Text Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to edit XML content from the XML tree in a text editor.



Modify the XML string and click OK. Your string will replace all sub-nodes of the node you selected in the XML tree. Note that you cannot modify the root element name.

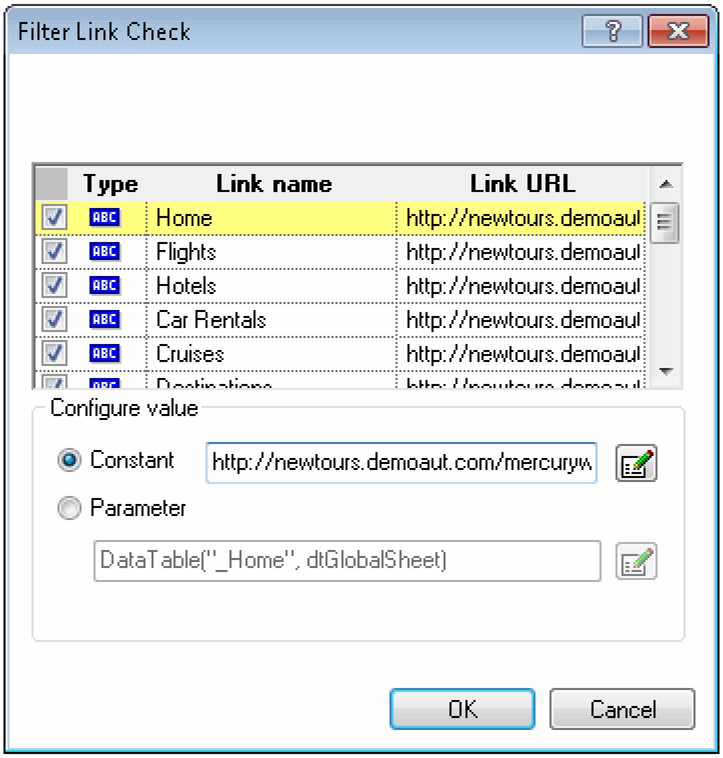
```
<TypeInformation Load="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xs
  <!-- DelphiTrackBar definition -->
  <ClassInfo Name="DelphiTrackBar" BaseClassInfoName="DelphiObject" GenericTypeID="o
    <IconInfo IconFile="INSTALLDIR\Dat\Extensibility\Delphi\Toolkits\DelphiExtSample\res\T
    <!-- Test object method definitions -->
    <TypeInfo>
      <!-- Definition of the Next test object method -->
      <Operation Name="Next" PropertyType="Method">
        <!-- Appears as tooltip in Keyword View -->
        <Description>Moves the track thumb to the next position.</Description>
        <!-- Used to generate the 'Documentation' column in Keyword View -->
        <Documentation><![CDATA[Move the track thumb to the next position on the %l %t.]]></I
      </Operation>
      <!-- Definition of the Prev test object method -->
      <Operation Name="Prev" PropertyType="Method">
        <!-- Appears as tooltip in Keyword View -->
        <Description>Moves the track thumb to the previous position.</Description>
        <!-- Used to generate the 'Documentation' column in Keyword View -->
        <Documentation><![CDATA[Move the track thumb to the previous position on the %l %t.]
```

To access	Click the **Edit XML as Text** button in the XML tree area of the XML Checkpoint Properties Dialog Box described on page 1432.
Important information	- This dialog box is used mainly for constructing an entire XML segment from a string or for fixing syntax problems that prevent the dialog box from displaying the XML tree correctly. It is also useful when you want to use copy-paste functionality to edit the tree. - When you click **OK** in the Edit XML as Text dialog box, the sub-tree of the node you previously selected in the XML tree (or the entire tree if no node was selected or if the root node was selected) is completely replaced by the XML content from the Edit XML as Text dialog box. - You cannot modify the name of the root element displayed in the Edit XML as Text dialog box.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419

Filter Link Check Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to instruct UFT to filter the hypertext links to check in a page checkpoint.



	Type	Link name	Link URL
☑	ABC	Home	http://newtours.demoaut
☑	ABC	Flights	http://newtours.demoaut
☑	ABC	Hotels	http://newtours.demoaut
☑	ABC	Car Rentals	http://newtours.demoaut
☑	ABC	Cruises	http://newtours.demoaut
☑	ABC	Destinations	http://newtours.demoaut

Configure value

● Constant http://newtours.demoaut.com/mercuryw

○ Parameter

DataTable("_Home", dtGlobalSheet)

To access	In the All Objects in Page Area area of the Page Checkpoint Properties Dialog Box (described on page 1432), click the **Filter Link Check** button.
Important information	If you select the **Links** check box in the Page Checkpoints Properties dialog box, all of the links on the page are automatically selected by default.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
See also	"Filter Image Check Dialog Box" on the next page

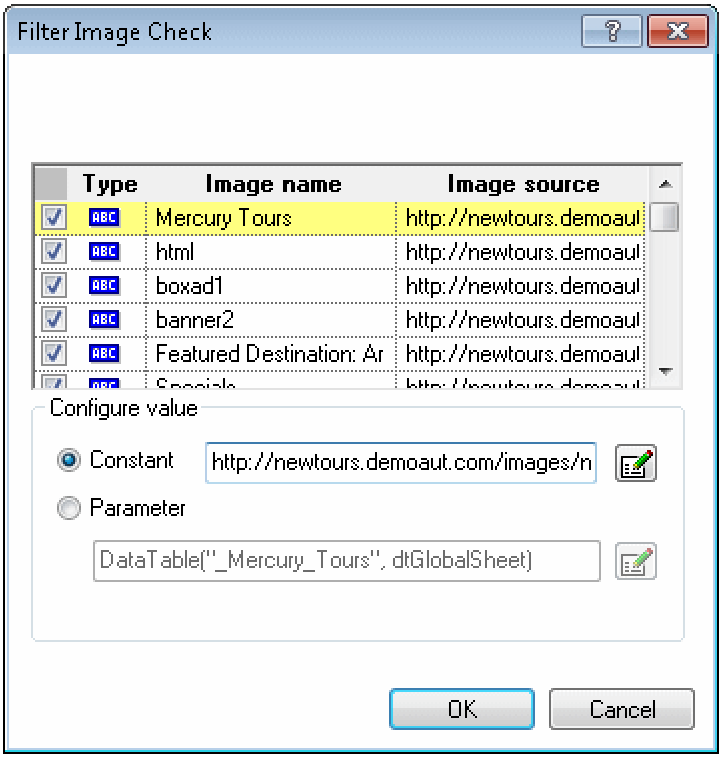
User interface elements are described below:

UI Element	Description
Check box	Each link on the page has a corresponding check box. To check a link, select the corresponding check box (by default all links are selected). To exclude a link from the page checkpoint, clear the corresponding check box.
Type	The ABC icon indicates that the target URL is currently a constant. The ▦ icon indicates that the value of the property is currently a DataTable parameter. The ✳ icon indicates that the value of the property is currently an environment variable parameter. The ● icon indicates that the value of the property is currently a random number parameter.
Link name	The text in the hypertext link.
Link URL	The target URL.
Configure Value area	Enables you to you define the expected value of the target URL to which the hypertext links as a **Constant** or **Parameter**. For details on the user interface elements in this area, see "Configure Value Area" on page 1455.

Filter Image Check Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to instruct UFT to filter which image sources to check in a to check in a page checkpoint.



	Type	Image name	Image source
☑	ABC	Mercury Tours	http://newtours.demoaut
☑	ABC	html	http://newtours.demoaut
☑	ABC	boxad1	http://newtours.demoaut
☑	ABC	banner2	http://newtours.demoaut
☑	ABC	Featured Destination: Ar	http://newtours.demoaut
☑	ABC	Specials	http://newtours.demoaut

Configure value

● Constant http://newtours.demoaut.com/images/n

○ Parameter

DataTable("_Mercury_Tours", dtGlobalSheet)

To access	In the Configure Value Area area of the Page Checkpoint Properties Dialog Box (described on page 1432), click the **Filter Image Check** button.
Important information	If you select the **Images** check box in the Page Checkpoints Properties dialog box, then by default all the images on the page are selected.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
See also	"Filter Link Check Dialog Box" on page 1470

User interface elements are described below:

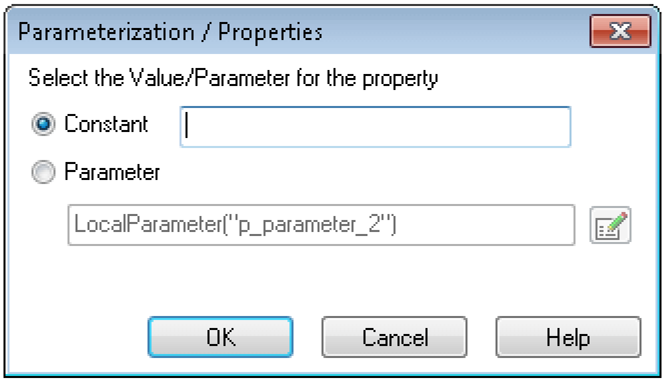
UI Element	Description
Check box	Each image source on the page has a corresponding check box. To check an image source, select the corresponding check box (by default all image sources are selected). To exclude an image source from the page checkpoint, clear the corresponding check box.
Type	The ABC icon indicates that the target URL is currently a constant. The ▦ icon indicates that the value of the property is currently a DataTable parameter. The icon indicates that the value of the property is currently an environment variable parameter. The icon indicates that the value of the property is currently a random number parameter.
Image name	The name of the image.
Image source	The image source file and path.
Configure Value area	Enables you to you define the path of the image source file as a **Constant** or **Parameter**. For details on the user interface elements in this area, see "Configure Value Area" on page 1455.

Parameterization / Properties Dialog Box (Checkpoints)

Relevant for: Keyword GUI components only

This dialog box enables you to set the checkpoint property value as a **Constant** or a **Parameter**.

The following image shows this dialog box when a **Constant** property value is selected.



Parameterization / Properties

Select the Value/Parameter for the property

● Constant

○ Parameter

LocalParameter("p_parameter_2")

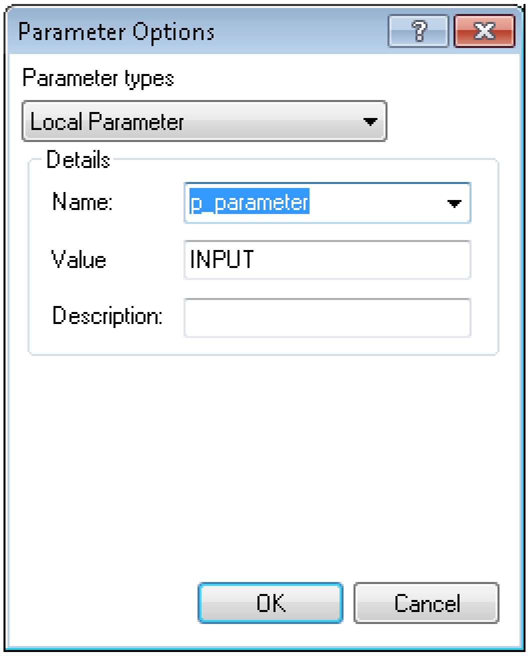
To access	In the "Checkpoint Properties Dialog Box" on page 1432 (simple mode) (described on page 1432), click the browse button ... for a property.
Important information	This dialog box opens only from Simple Mode. (In Advanced Mode, the options in this dialog box are available in the **Configure value** area of the Checkpoint Properties dialog box.)
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
See also	"Checkpoint Properties Dialog Box" on page 1432

For a description of user interface elements, see "Configure Value Area " on page 1575.

Parameter Options Dialog Box (Local and Component Input Parameters for Checkpoints)

Relevant for: GUI components

This dialog box enables you to set the parameter value for a checkpoint property value.



Parameter Options

Parameter types: Local Parameter

Details

Name: p_parameter

Value: INPUT

Description:

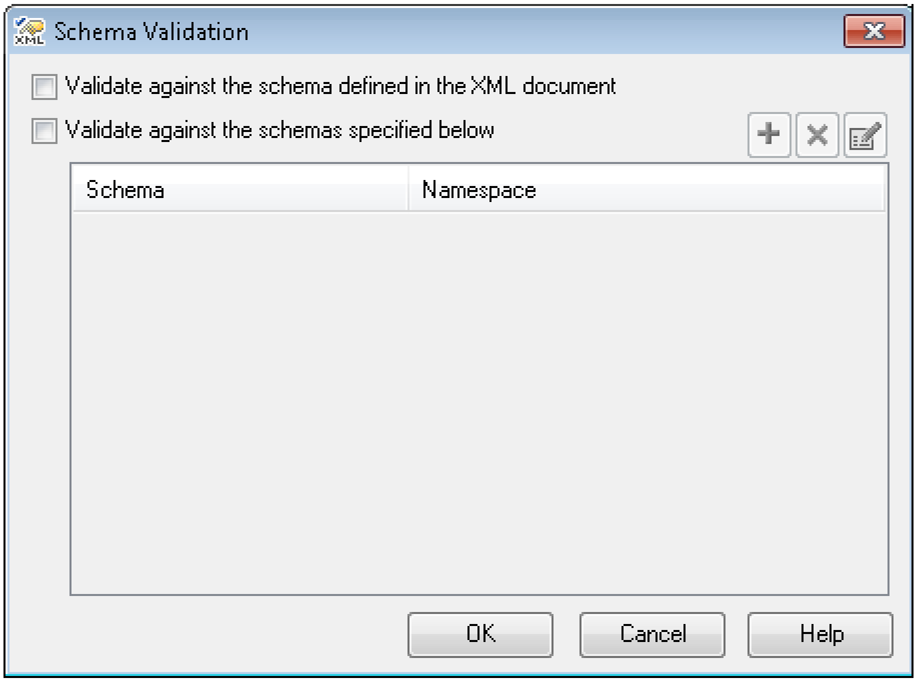
To access	• In the "Checkpoint Properties Dialog Box" on page 1432 (described on page 1432), click the **Parameter Options** button . • In the "Parameterization / Properties Dialog Box (Checkpoints)" (described on page 1473), select the **Parameter** radio button. If a parameter is already defined, you must then click the **Parameter Options** button to open this dialog box.
Relevant tasks	• "How to Insert a Checkpoint in a GUI Test or Component" on page 1419 • "How to Parameterize Input Values (Keyword Components)" on page 1573
See also	• "Checkpoint Properties Dialog Box" on page 1432 • "Value Configuration Options Dialog Box" on page 1579

For a description of user interface elements, see "Configure Value Area " on page 1575.

Schema Validation Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to specify an XML schema against which you want to validate the hierarchy of the XML in your application or file.



□ Validate against the schema defined in the XML document

□ Validate against the schemas specified below

To access	In the "Checkpoint Properties Dialog Box" (described on page 1432), select the **Activate Schema Validation** button.

Important information	• If you are validating an XML file using a schema defined in the XML file, the schema can be defined with an absolute or relative path. When you specify a relative path, UFT searches for the schema in the folders listed in the **Folders** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Folders** node). For details, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544. • If you are validating an XML document located on the Web with a schema file located on your file system, you cannot use UNC format (for example, \\ComputerName\Path\To\Schema) to specify the schema file location. Instead, map the schema file location to a network drive. • If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document. Using an external XML schema file to validate an XML document may cause an unexpected result if the XML document has an XML schema declaration, and the namespace in the external schema file and the schema defined in the document are not identical. • When you perform a schema validation, UFT validates all of the elements in the XML document, even if certain XML elements are not associated with a schema file. Any XML elements that are not associated with a schema file cause the schema validation to fail.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419

User interface elements are described below:

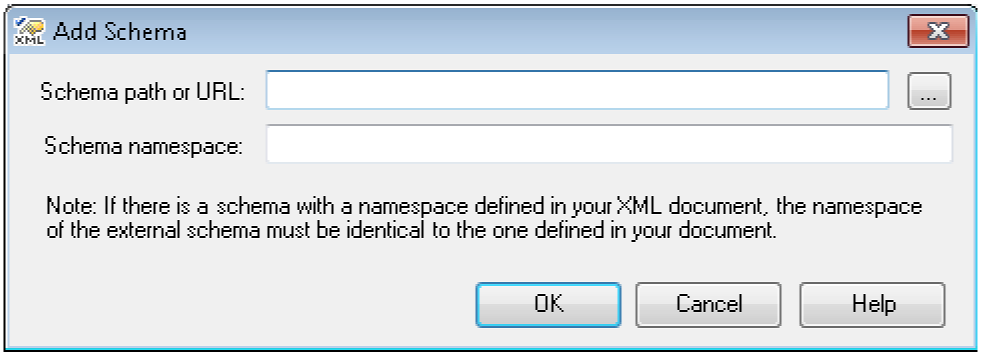
UI Elements	Description
Validate against the schema defined in the XML document	Instructs UFT to use the schema or schemas defined within your XML document to validate the hierarchy of the XML in your Web page/frame, XML file, or XML test object.
Validate against the schemas specified below	Instructs UFT to use one or more external XML schema files to validate the hierarchy of your XML. Any schemas defined within your XML document are also checked. **Note:** If you select this option: • The **Validate against the schema defined in the XML document** option is automatically selected and is disabled. • The **Add Schema**, **Remove Schema**, and **Modify Schema** buttons are enabled.
+	**Add Schema.** Enables you to add an external schema file to the list. For details, see "Add Schema Dialog Box" on the next page.

UI Elements	Description
✕	**Remove Schema.** Enables you to remove the selected external schema file from the list.
📝	**Modify Schema.** Enables you to modify the details of the selected external schema file in the list. For details, see "Edit Schema Dialog Box" on the next page.

Add Schema Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to specify the path or URL of an external schema file and its namespace. If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document.



Note: If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document.

To access	In the "Schema Validation Dialog Box" (described on page 1476), select the **Add Schema** button
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419

User interface elements are described below:

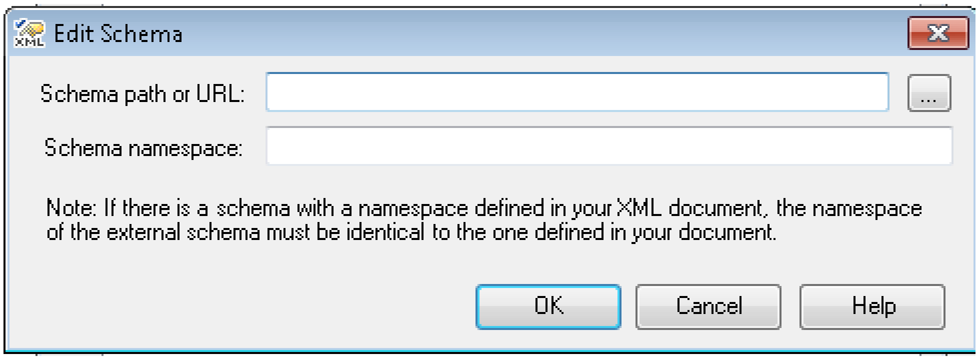
UI Elements	Description
Schema path or URL	The path or URL of your XML schema file. Alternatively, click the browse button to navigate to the XML schema you want to use to validate the XML in your Web page/frame, XML file, or XML test object. You can specify schema files either from your file system or from ALM. For each external file you add, you must specify its path or URL and namespace

UI Elements	Description
Schema namespace	(If applicable.) The namespace of your schema file. UFT checks that the namespace matches the schema file as part of the validation process. If the schema file has a namespace and you do not specify it, or if the namespace you specify is different to the one specified in the schema file, the validation fails. Click **OK** to add the selected schema to the list in the Schema Validation dialog box. Click the **Add Schema** button again if you want to add another schema.

Edit Schema Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box displays the path and namespace of the schema file you selected in the list. You can modify the path or URL of the selected schema file, and its namespace.



Note: If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document.

To access	In the "Schema Validation Dialog Box" (described on page 1476), select the **Modify Schema** button
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419

User interface elements are described below:

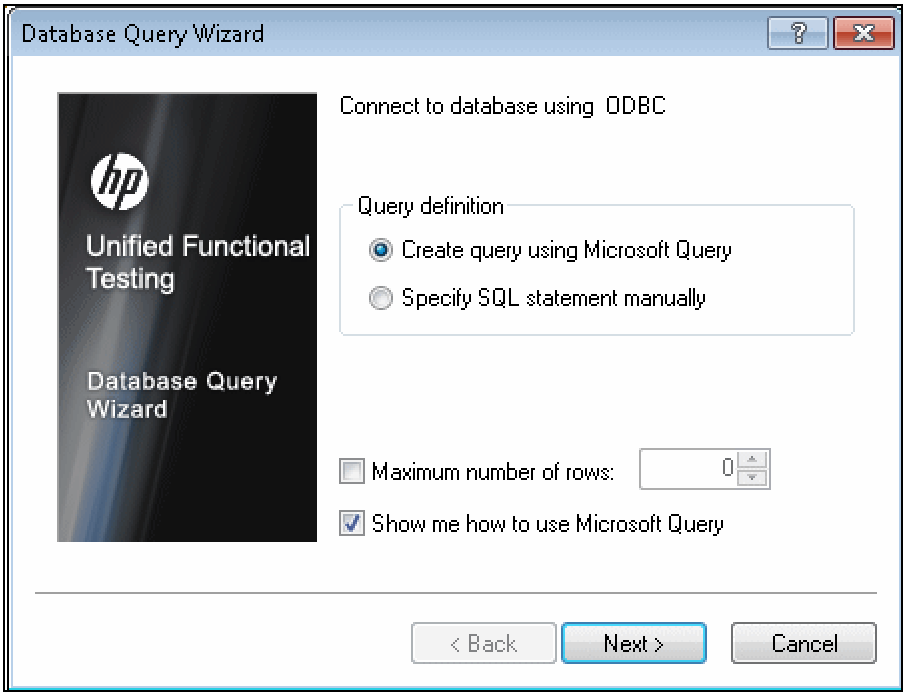
UI Elements	Description
Schema path or URL	The path or URL of your XML schema file. Alternatively, click the browse button to navigate to the XML schema you want to use to validate the XML in your Web page/frame, XML file, or XML test object. You can specify schema files either from your file system or from ALM. For each external file you add, you must specify its path or URL and namespace

UI Elements	Description
Schema namespace	(If applicable.) The namespace of your schema file. UFT checks that the namespace matches the schema file as part of the validation process. If the schema file has a namespace and you do not specify it, or if the namespace you specify is different to the one specified in the schema file, the validation fails. Click **OK** to add the selected schema to the list in the Schema Validation dialog box. Click the **Edit Schema** button again if you want to edit another schema.

Connect to Database Using ODBC Page (Database Query Wizard)

Relevant for: GUI tests and scripted GUI components

This wizard enables you to define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement.



Connect to database using ODBC

Query definition

● Create query using Microsoft Query

○ Specify SQL statement manually

□ Maximum number of rows: 0

☑ Show me how to use Microsoft Query

To access	1. Ensure that a GUI action or component is in focus in the document pane. 2. Select **Design > Checkpoint > Database Checkpoint** .
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
Wizard map	This wizard contains: **Connect to Database Using ODBC** > **Specify SQL Statement**(page 1481)
See also	"Specify SQL Statement Page (Database Query Wizard)" on the next page

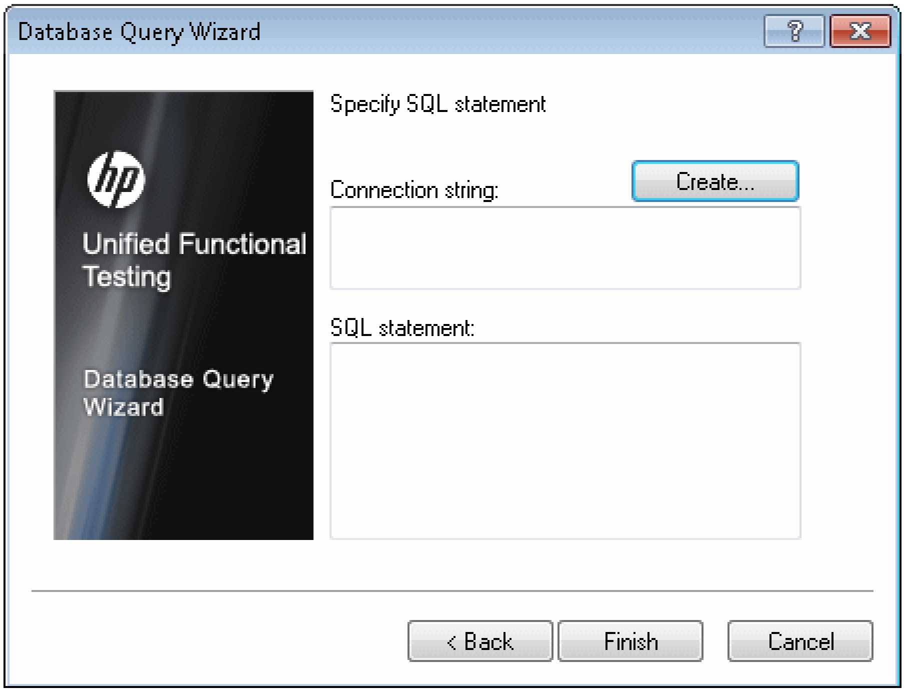
User interface elements are described below:

UI Elements	Description
Create query using Microsoft Query	Opens Microsoft Query, enabling you to create a new query. After you finish defining your query, you return to UFT. This option is available only if you have Microsoft Query installed on your computer.
Specify SQL statement manually	Opens the **Specify SQL statement** page in the wizard, which enables you to specify the connection string and an SQL statement.
Maximum number of rows	When selected, enables you to limit the number of rows and enter the maximum number of database rows to check. You can specify a maximum of 32,000 rows.
Show me how to use Microsoft Query	Displays an instruction page when you click **Next** before opening Microsoft Query. Available only when **Create query using Microsoft Query** is selected.

Specify SQL Statement Page (Database Query Wizard)

Relevant for: GUI tests and scripted GUI components

This page enables you to manually specify the database connection string and the SQL statement.



To access	Select **Design > Checkpoint > Database Checkpoint**.

Important information	It takes several seconds to capture the database query and restore the UFT window.
Relevant tasks	"How to Insert a Checkpoint in a GUI Test or Component" on page 1419
Wizard map	This wizard contains: **Connect to Database Using ODBC** (page 1480) > **Specify SQL Statement**
See also	"Connect to Database Using ODBC Page (Database Query Wizard)" on page 1480

User interface elements are described below:

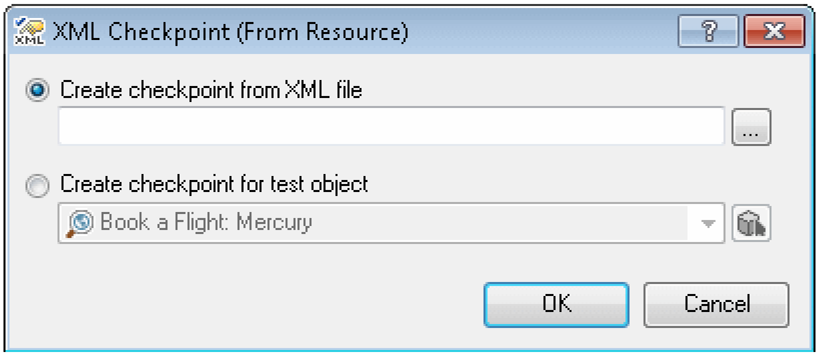
UI Elements	Description
Connection string	The connection string.
Create	Opens the ODBC Select Data Source dialog box. You can select a .dsn file in the ODBC Select Data Source dialog box or create a new .dsn file to have the Database Query Wizard insert the connection string in the box for you.
SQL statement	The details of the SQL statement.

XML Source Selection - Checkpoint / Output Value Properties Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to specify the XML file or test object for which you want to insert an XML checkpoint/output value step.

The following image shows an example of the XML Source Selection - Checkpoint Properties dialog box.



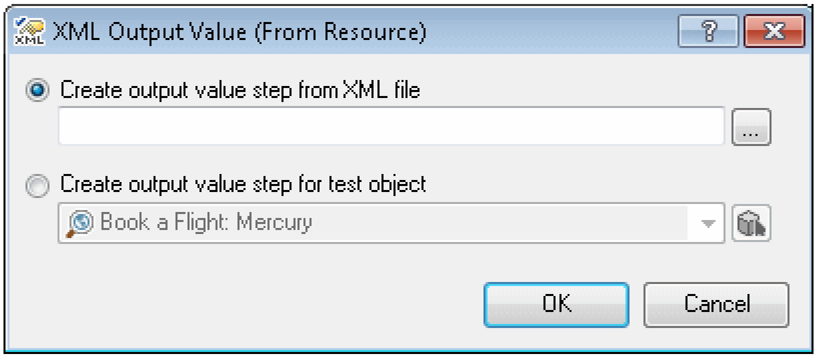
XML Checkpoint (From Resource)

● Create checkpoint from XML file

○ Create checkpoint for test object

Book a Flight: Mercury

The following image shows an example of the XML Source Selection - Output Value Properties dialog box.



XML Output Value (From Resource)

● Create output value step from XML file

○ Create output value step for test object

Book a Flight: Mercury

To access	1. Ensure that a GUI action or component is in focus in the document pane. 2. Use one of the following: ▪ **For checkpoints:** Select **Design > Checkpoint > XML Checkpoint (From Resource)** ▪ **For output values:** Select **Design > Output Value > XML Output Value (From Resource)**
Relevant tasks	• "How to Insert a Checkpoint in a GUI Test or Component" on page 1419 • "How to Create or Modify an Output Value Step" on page 1494

User interface elements are described below:

UI Elements	Description
Create checkpoint step from XML file / Create output value step from XML file	Specifies the file for which you want to insert a checkpoint/output value. In the box, you can: • Enter the URL or the file path of the XML file. • Click the browse button to open the Open XML File dialog box and navigate to the XML file. You can specify an XML file either from your file system or from ALM. **Note:** If you enter a relative path, ALM searches for the XML file in the folders listed in the Folders pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Folders** pane). After ALM locates the file, it saves it as an absolute path and uses the absolute path during the run session. For details, see "Folders Pane (Options Dialog Box > GUI Testing Tab)" on page 544.

UI Elements	Description
Create checkpoint step for test object / Create output value step for test object	- Specifies an XML checkpoint/output value for a test object. From the list, select the test object. - You can select an existing WebXML or XMLFile test object type as long as the actual XML object is currently available (in an open browser or in the file system, as relevant). **Note:** Selecting a WebXML or XMLFile test object is identical to using the **XML Checkpoint (From Application)** or **Create new checkpoint from file** options, but may be faster than browsing to these objects and can be inserted while recording or editing. However, to use this option, the XML source must be available when you select the test object (the Web page must be open or the file must exist in the same location as when the test object was defined). **Tip:** To select an object that is not displayed in the list, you can click **Object from Repository** . Then select a test object from the object repository on which to create a new checkpoint. The selected object must support XML (such as a **WebXML** test object). For details on selecting objects, see "Select Test Object Dialog Box" on page 958.

Troubleshooting and Limitations - Using Checkpoints

Relevant for: GUI tests and components

The following sections describe troubleshooting and limitations for working with checkpoints in a GUI test or component:

Accessibility checkpoints

In ALM, you can view a comparison of accessibility checkpoints in the Asset Comparison Tool only if both UFT and the UFT Add-in for ALM are installed on the ALM computer.

Bitmap checkpoints

Bitmap checkpoints on objects containing text may fail if you create them using a Remote Desktop Connection and then run them locally, or if you create them locally and then run the checkpoint steps using a Remote Desktop Connection. In the run results, the image displayed when you click **View Difference** in the bitmap checkpoint results shows some text shapes.

Workaround: Enable the **Font smoothing** option in the Remote Desktop Connection application.

Database checkpoints

- The format of captured values varies depending on the specific system settings. For example, date and time values may be set to different formats.

  Workaround: If you are running the test or scripted component on a different system than the one you used to record the test, confirm that the systems use the same format settings.

- When you create a database checkpoint on one machine and try to run it on different machine, you should have the same ODBC driver installed on both machines.

File content checkpoints

File content checkpoints for htm/html files generated dynamically by third-party Javascript code are not supported.

Text/text area checkpoints

Text and text area checkpoints are not supported if the text in the selected area is in a non-English language.

XML checkpoints

- When executing an XML checkpoint on an XML file that contains **>** as a value, an error message may occur.

- When you add a new value node to an XML node, in some cases the new value may not be displayed.

  Workaround: Close the Edit XML as Text dialog box and reopen it to display the new value node correctly.

- When inserting an XML file checkpoint on a file that cannot be loaded, or a file that is formatted incorrectly, you may receive an error message.
- Creating and running XML checkpoints for large XML documents may take a few minutes.

Chapter 51: Output Values in GUI Testing

Relevant for: GUI tests and components

This chapter includes:

Concepts

Output Values Overview

Relevant for: GUI tests and components

Your test or component can retrieve values and store them in output value objects. You can then use these values as input at a later stage in a run session.

An **output value** step is a step in which one or more values are captured at a specific point in your test or component and stored for the duration of the run session. The values can later be used as input at a different point in the run session.

You can output the property values of any object. You can also output values from text strings, table cells, databases, and .xml documents.

When you create output value steps, you can determine where the values are stored during the run session and how they can be used. During the run session, UFT retrieves each value at the specified point and stores it in the specified location. When the value is needed later in the run session, UFT retrieves it from this location and uses it as required.

Output values are stored only for the duration of the run session. When the run session is repeated, the output values are reset.

> **Note:** After the run session, you can view the output values retrieved during the session as part of the session results. For details, see the section on Parameterized Values in the Run Results (described in the *HP Run Results Viewer User Guide*).

For details about using output values for each add-in environment installed with UFT, see "Supported Output Values" on page 2228.

This section also includes:

Output Value Categories

Relevant for: GUI tests and components

You can create the following categories of output values:

- "Standard Output Values" below
- "File Content Output Values (tests and scripted components only)" below
- "Table Output Values (tests and scripted components only)" on the next page
- "Text and Text Area Output Values (tests and scripted components only)" on the next page
- "Database Output Values (tests and scripted components only)" on the next page
- "XML Output Values (tests and scripted components only)" on the next page
- "Existing Output Values (tests and scripted components only)" on page 1491

Standard Output Values

You can use standard output values to output the property values of most objects. For example, you can use standard output values to output text strings by specifying the **text** property of the object as an output value in a keyword component. In a Web-based application, the number of links on a Web page may vary based on the selections a user makes on a form on the previous page. You could create an output value in your test or scripted component to store the number of links on the page.

> **Note for tests and scripted components:**
>
> - You can also use standard output values to output the contents of table cells.
> - You can use standard output values to output text strings by specifying the **text** property of the object as an output value.

For task details, see "How to Create or Modify an Output Value Step" on page 1494.

File Content Output Values (tests and scripted components only)

You can use file content output values to output the contents from any of the following file types:

• HTML	• Microsoft Word	• Text
• PDF	• RTF	

You can create output values from the entire contents of a file, or from a part of it. During the run session, UFT retrieves the current data from the file and outputs the values according to the settings that you specified.

For task details, see "How to Create or Modify an Output Value Step" on page 1494.

Table Output Values (tests and scripted components only)

Table output values are a subset of standard output values, described above. You can use table output values to output the contents of table cells. For some types of tables, you can specify a row range from which to choose the table cells. During the run session, UFT retrieves the current data from the specified table cells according to the settings that you specified and outputs the values to the Data pane.

For task details, see "How to Create or Modify an Output Value Step" on page 1494.

Text and Text Area Output Values (tests and scripted components only)

You can use text output values to output text strings displayed in an application. When creating a text output value, you can output a part of the object's text. You can also specify the text before and after the output text.

You can use text area output values to output text strings displayed within a defined area of a screen in a Windows-based application.

For example, suppose that you want to store the text of any error message that appears after a specific step in the Web application you are testing. Inside the **If** statement, you check whether a window exists with a known title bar value, for example Error. If it exists, you output the text in this window (assuming that the window size is the same for all possible error messages).

> **Note:** You can create a text area output value only while recording on Windows-based applications.

For task details, see "How to Create or Modify an Output Value Step" on page 1494.

Database Output Values (tests and scripted components only)

You can use database output values to output the value of the contents of database cells, based on the results of a query (result set) that you define on a database. You can create output values from the entire contents of the result set, or from a part of it. During the run session, UFT retrieves the current data from the database and outputs the values according to the settings that you specified.

For task details, see "How to Create or Modify an Output Value Step" on page 1494.

XML Output Values (tests and scripted components only)

You can use XML output values to capture and output the values of XML elements and attributes in XML documents.

For example, suppose that an XML document in a Web page contains a price list for new cars. You can output the price of a particular car by selecting the appropriate XML element value to output.

After the run session has finished, you can also view the captured data by opening the XML Output Value Results window from the Run Results Viewer. For details, see the XML Output Value Results Window (described in the *HP Run Results Viewer User Guide*).

For task details, see "How to Create or Modify an Output Value Step" on page 1494.

Existing Output Values (tests and scripted components only)

When you insert an existing output value in your test or scripted component, consider which output values should be used in multiple locations in your test or component. Each time an output value step is performed, the value contained in the output value is overwritten with the new output value. You should insert an existing output value only if the stored value will no longer be needed later in the run session when the output value object is used again.

For details, see "Add Existing Output Value Dialog Box" on page 1515.

Output Types and Settings

Relevant for: GUI tests and components

In the "Output Options Dialog Box " (described on page 1519) you can set the output type and settings for each value, to determine where it is stored and how it can be used during the run session. When the output value step is reached, UFT retrieves each value selected for output and stores it in the specified location for use later in the run session.

For details relevant to keyword components, see "Output Value Properties Dialog Box" on page 1498.

When you create a new output value step, UFT assigns a default definition to each value selected for output. For details, see the Default Output Definitions (tests only) below.

You can change the current output definition for the selected value by selecting a different output type and/or changing the output settings in the "Output Options Dialog Box "(described on page 1519).

Default Output Definitions (tests only)

When you initially select a value for output, UFT generates a default output definition for the value.

When you output a value for a step, the following occurs:

- If at least one output parameter is defined in the action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the "Parameters Tab (Action Properties Dialog Box)" (described on page 903).

- If no output parameters are defined in the action, the default output type is **DataTable**, and UFT creates a new DataTable output name based on the selected value.

  The output value is created in the Global sheet of the Data pane.

  For details on creating output parameters for actions, see "How to Display and Modify Action-Related Information" on page 886.

  For details on Data pane sheets, see "Data Pane" on page 221.

Viewing and Editing Output Values

Relevant for: GUI tests and components

In the Keyword View, an output value step is displayed as follows:

- The **Operation** column displays Output.
- The **Value** column displays CheckPoint followed by the name assigned to the output value.

In the Editor, an output value statement is displayed with the following syntax:

```
Object.Output CheckPoint(Name)
```

You can view or edit the output value or its details in the relevant Output Value Properties dialog box, by right-clicking the step and choosing **Output Value Properties**. Alternatively, you can click the step in the **Value** column in the Keyword View and then click the **Output Properties** button .

For details on the options available in the various Output Value Properties dialog boxes, see "Output Value Properties Dialog Box" on page 1498.

Storing Output Values

Relevant for: GUI tests and scripted GUI components

When you define an output value, you can specify where and how each value is stored during the run session.

You can output a value to:

- "Test and Action Parameters (tests only)" below
- "Run-time Data Table"
- "Environment Variables"

Test and Action Parameters (tests only)

You can output a value to an action parameter, so that values from one part of a run session can be used later in the run session, or be passed back to the application that ran (called) the test.

For example, suppose you are testing a shopping application that calculates your purchases and automatically debits your account with the amount that you purchased. You want to test that the application correctly debits the purchase amount from the account each time that the action is run with a different list of items to purchase. You could output the total amount spent to an action parameter value, and then use that value later in your run session in the action that debits the account.

For details on action parameters, see "Action Parameters" on page 877.

Run-time Data Table

The option to output a value to the run-time data table is especially useful with a **data-driven** test, action, or scripted component that runs several times. In each repetition, or **iteration**, UFT retrieves the current value and stores it in the appropriate row in the run-time data table.

Example

Suppose you are testing a flight reservation application and you design a test to create a new reservation and then view the reservation details. Every time you run the test, the application generates a unique order number for the new reservation. To view the reservation, the application requires the user to input the same order number. You do not know the order number before you run the test.

To solve this problem, you output a value to the Data pane for the unique order number generated when creating a new reservation. Then, in the View Reservation screen, you use the column containing the stored value to insert the output value into the order number input field.

When you run the test, UFT retrieves the unique order number generated by the site for the new reservation and enters this output value in the run-time data table. When the test reaches the order number input field required to view the reservation, UFT inserts the unique order number stored in the run-time data table into the order number field.

Environment Variables

When you output a value to an internal user-defined environment variable, you can use the environment variable input parameter at a later stage in the run session.

Example

Suppose you are testing an application that prompts the user to input an account number on a Welcome page and then displays the user's name. You can use a text output value to capture the value of the displayed name and store it in an environment variable.

You can then retrieve the value in the environment variable to enter the user's name in other places in the application. For example, in an Order Checkbook Web page, which for security reasons requires users to enter the name to appear on the checks, you could use the value to insert the user's name into the **Name** edit box.

Note:

- Output values are stored only for the duration of the run session, and are not saved with the test or component. If you select to output a value to an existing parameter, a data table column, or an environment variable, the existing value is overwritten when the output value step runs. When the run session ends, the original value is restored.
- You can output values only to internal user-defined environment variables and not to external or built-in environment variables, which are read-only.

Tasks

How to Create or Modify an Output Value Step

Relevant for: GUI tests and components

This task describes how to insert an output value step while recording or editing your test or component. You can also add an existing output value to a test or scripted component. It is generally more convenient to define output values after creating the initial test or component.

For details on supported output values per add-in environment, see "GUI Checkpoints and Output Values Per Add-in" on page 2224

This task describes both the general process for inserting a new output value step to your test or component and the prerequisites and considerations for the different types of output values.

This task includes the following steps:

- "Important information before inserting the output value step" below
- "Insert a new standard output value step while recording your test or component" on the next page
- "Insert a new output value step using the Active Screen while editing (tests and scripted components only)" on page 1496
- "Insert an existing output value step while editing (tests and scripted components only)" on page 1497
- "Set the options for the output value object" on page 1497

Important information before inserting the output value step

Object visibility in application	• During an editing session, make sure the object is visible in your application before inserting an output value step on it. • **For file content output values:** The source file must be located on the file system.
Availability	• Recording sessions • Editing sessions • Active Screen
Active Screen	Make sure that the Active Screen contains sufficient data for the object for which you want to define an output value. For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

Important Information	• Output valuescan be viewed in the following modes: ▪ **Simple Mode:** Displays only the basic properties and expected values of the output value. ▪ **Advanced Mode:** Displays all supported properties and expected values of the output value. • **For text/text area checkpoints:** ▪ Before you create a text or text area output value for a Windows-based application, make sure you configure the required capture settings in the **Text Recognition** pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Text Recognition** node). For details, see "Text Recognition Pane (Options Dialog Box > GUI Testing Tab)" on page 542. ▪ When you use text-area selection to capture text displayed in a Windows application, it is often advisable to define a text area larger than the actual text you want UFT to use as an output value. During the run session, UFT outputs the selected text, within the defined area, according to the settings you configured. ▪ Because text may change its position during run sessions, make sure that the area defined is large enough so that the output text is always within its boundaries. For details, see "Text Recognition Overview" on page 1408.

Insert a new standard output value step while recording your test or component

1. Start a recording session.

2. Do one of the following:

   - Select **Design > Output Value > Standard Output Value**.

   - In the record toolbar, click **Insert Checkpoint or Output Value** down arrow and select **Standard Output Value**.

     The pointer changes into a pointing hand. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 1196.

   If necessary, select the object or object sections you want to include in the output value. The specific dialog that opens differs depends on the type of output value selected:

   - **For table checkpoints**, see "Define/Modify Row Range Dialog Box".

   - **For database checkpoints**, see "Connect to Database Using ODBC Page (Database Query Wizard)" on page 1480.

- **For XML checkpoints**, see "XML Source Selection - Checkpoint / Output Value Properties Dialog Box".

Insert a new output value step using the Active Screen while editing (tests and scripted components only)

1. Do one of the following:

   - Select **View > Active Screen**, click a step whose Active Screen contains the object for which you want to specify an output value, and right-click the object for which you want to specify an output value and select **Insert Output Value**.

   - Right-click the step and select **Insert Output Value**.

     If the location you clicked is associated with more than one object, the "Object Selection Dialog Box " (described on page 1195) opens.

     If necessary, select the object or object sections you want to include in the output value. The specific dialog that opens differs depends on the type of output value selected:

     - **For table checkpoints**, see "Define/Modify Row Range Dialog Box".

     - **For database checkpoints**, see "Connect to Database Using ODBC Page (Database Query Wizard)" on page 1480.

     - **For XML checkpoints**, see "XML Source Selection - Checkpoint / Output Value Properties Dialog Box".

Note if you are creating a text/text area output value:

- To output text value while editing, you first highlight a text string in the Active Screen then right-click the string, and select **Insert Text Output Value**.

- When you output a text area value, you first define the area containing the text you want UFT to check. When you select Text Area Output Value, the mouse turns into a crosshairs pointer. Click and drag the crosshairs pointer to define this area. Release the mouse button after outlining the area required. For more details, see the section on text area output values in "Output Value Categories" on page 1489, and the important details in "Output Value Properties Dialog Box" on page 1498.

Tip: Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

Insert an existing output value step while editing (tests and scripted components only)

1. Select the step after which you want to insert the output value.

2. Select **Design > Output Value > Existing Output Value**. The Add Existing Output Value Dialog Box opens, enabling you to select the test object from which you want to output values.

Set the options for the output value object

For tests and scripted components: Set the options for the output value, including the values to add to the output value step. For details, see "Output Value Properties Dialog Box" on the next page.

For keyword components: Specify the settings in the "Output Value Properties Dialog Box" on the next page (described on page 1498).

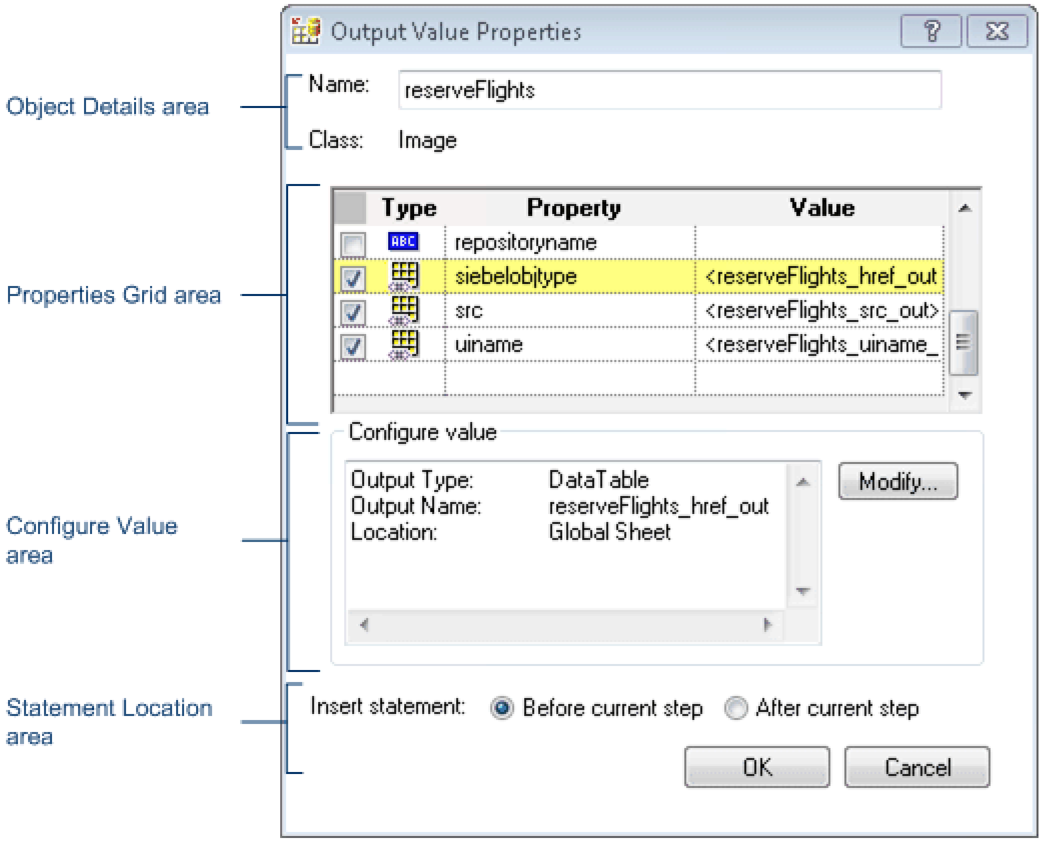
Reference

Output Value Properties Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to define and modify properties for a selected output value.

The example below gives the basic structure for the various Output Value Properties dialog boxes. Certain elements may differ between output value types.



Name: reserveFlights

Class: Image

	Type	Property	Value
□	ABC	repositoryname	
☑	▦	siebelobjtype	<reserveFlights_href_out
☑	▦	src	<reserveFlights_src_out>
☑	▦	uiname	<reserveFlights_uiname_

Output Type: DataTable
Output Name: reserveFlights_href_out
Location: Global Sheet

Insert statement: ● Before current step ○ After current step

To access	1. Ensure that a GUI action or component is in focus in the document pane. 2. Use one of the following: ▪ Insert a new output value step and select an object from your application. ▪ During a recording session, click the **Insert Checkpoint or Output Value** down arrow and select the relevant type. ▪ Right-click an existing output value step and select **Output Value Properties**. ▪ In the local or shared object repository, click an existing output value object. The output value details are displayed on the right side of the object repository window, in the **Object Details** area.
Important information	• If you insert an output value on a Web page, the Page Output Value Properties dialog box opens. This dialog box is identical to the Output Value Properties dialog box, except that it contains two additional areas, **HTML verification** and **All objects in page**. These options are relevant only for checkpoints and are disabled when defining output values. • You cannot view or edit the advanced properties of output value objects when working in ALM. Therefore, if one or more advanced properties are selected in an output value object, and an ALM user views its properties in ALM, the dialog box displays text indicating that some properties are not shown.
Relevant tasks	"How to Create or Modify an Output Value Step" on page 1494

This dialog box has the following areas:

- "Object Details Area" below
- "Properties Grid Area" on the next page
- "Statement Location Area" on page 1501

Object Details Area

UI Element	Description
Name	The name that UFT assigns to the output value object. By default, the name is the same as the name of the object on which the output value step is performed. You can specify a different name for the output value object or accept the default name.

UI Element	Description
Class	The type of object (read-only). **Note:** The class of the object is not displayed in ALM when working with a keyword component.
	Find in Repository. Displays the output value object in its existing repository. For **tests and scripted components**, this option is only available when editing an existing output value step. It is not available when creating a new output value step. For **keyword components**, this option is only available in Advanced Mode.
(available only for keyword components)	**Advanced Mode/Simple Mode.** Toggles the dialog box between Advanced Mode and Simple Mode. Advanced Mode expands the dialog box and enables you to view or edit advanced properties.

Properties Grid Area

The Properties grid area displays the properties to output from the object.

For **standard output values**, see "Properties Grid Area (Output Value Properties Dialog Box) - Standard Output Values" on the next page.

For **table output values**, see "Properties Grid Area (Output Value Properties Dialog Box) - Table Output Values" on page 1507.

For **text or text area output values**, see "Output Value Summary Area (Output Value Properties Dialog Box) - Text/Text Area Output Values" on page 1509.

For **file content output values**, see "File and Output Value Details Area (Output Value Properties Dialog Box) - File Content Output Values" on page 1504.

For **database output values**, see "Properties Grid Area (Output Value Properties Dialog Box) - Database Output Values" on page 1503.

For XML output values, see "XML Tree/Options Area (Output Value Properties Dialog Box) - XML Output Values" on page 1512.

Configure Value Area

For **standard**, **table**, or **database** output values, see "Configure Value Area (Output Value Properties Dialog Box) - Standard, Table, and Database Checkpoints" on page 1502.

For **text/text area** output values, see "Output Value Summary Area (Output Value Properties Dialog Box) - Text/Text Area Output Values" on page 1509.

For *file content* or *XML* output values, this area is not applicable.

Statement Location Area

The following image shows the statement location area when inserting a new output value step during an editing session. When inserting a new output value step during a recording session, the **Insert statement** option is not available.

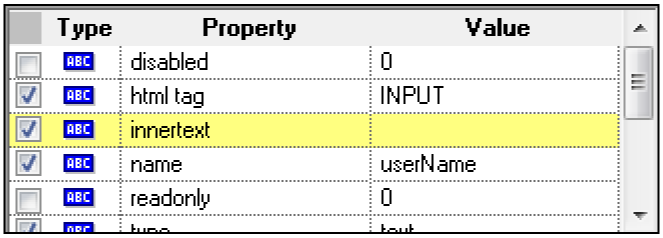
User interface elements are described below:

UI Elements	Description
Insert statement	Specifies whether to insert the output value step before or after the currently selected step. The default value is **Before current step**. **Note:** Available only when inserting a new output value step during an editing session. During a recording session, the output value step is always inserted as the next step.

Properties Grid Area (Output Value Properties Dialog Box) - Standard Output Values

Relevant for: GUI tests and components

This area of the Output Properties dialog box enables you to set the values to output for the output value step.



	Type	Property	Value
☐	ABC	disabled	0
☑	ABC	html tag	INPUT
☑	ABC	innertext	
☑	ABC	name	userName
☐	ABC	readonly	0
☑	ABC	type	text

User interface elements are described below:

UI Elements	Description
Check box	Enables you to select one or more property for the object, and specify the output options for each property value you select.

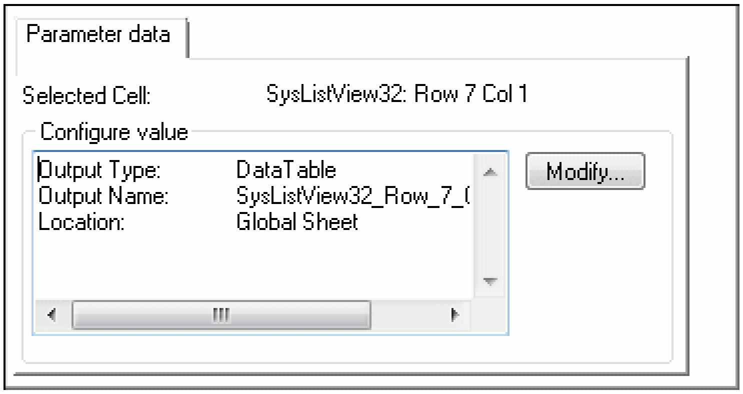
UI Elements	Description
Type	• ABC . Indicates that the value of the property is currently a constant. • . Indicates that the value of the property is currently a test or action parameter (tests only) • . Indicates that the value of the property is currently a DataTable parameter (tests and scripted components only) • . Indicates that the value of the property is currently an environment variable parameter (tests and scripted components only) • . Indicates that the value of the property is currently a random number parameter (tests and scripted components only) • # . Indicates that the value of the property is currently a parameter (keyword components only)
Property	The name of the property.
Value	The expected value of the property. For details on modifying the value of a property in tests and scripted components, see "Configure Value Area " on page 1575.

Configure Value Area (Output Value Properties Dialog Box) - Standard, Table, and Database Checkpoints

Relevant for: GUI tests and components

The Configure Value area enables you to set the value to use as an output.

The example below shows the Configure Value area (called the **Parameter data** tab) when a table output value is selected.



Selected Cell: SysListView32: Row 7 Col 1

Output Type: DataTable
Output Name: SysListView32_Row_7_(
Location: Global Sheet

Important information (for keyword components only)	- This area is available only in Advanced Mode. - In Simple Mode, when you click the **Browse** button for a selected property ... in the Properties grid, the "Parameterization / Properties Dialog Box (Output Values)" opens, in which the output definition for the selected property value is displayed.

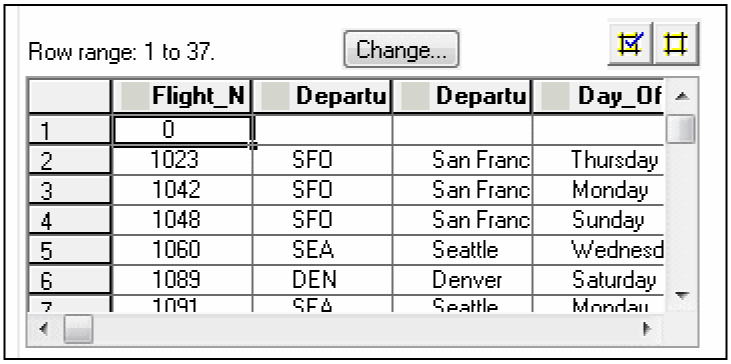
User interface elements are described below:

UI Elements	Description
Selected Cell (table and database checkpoints only)	The name of the object and location of the cell to output.
Selected Cell	The name of the object and location of the cell to output.
Modify	Opens the Output Options dialog box, enabling you to change the output type and/or settings for the selected value. For details, see "Output Options Dialog Box " on page 1519.

Properties Grid Area (Output Value Properties Dialog Box) - Database Output Values

Relevant for: GUI tests and scripted GUI components

For database output values, the Properties Grid area displays a grid containing the database values of the object for which you are creating an output value.



Row range: 1 to 37.

	Flight_N	Departu	Departu	Day_Of
1	0			
2	1023	SFO	San Franc	Thursday
3	1042	SFO	San Franc	Monday
4	1048	SFO	San Franc	Sunday
5	1060	SEA	Seattle	Wednesd
6	1089	DEN	Denver	Saturday
7	1091	SEA	Seattle	Monday

Important information	• Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the output are removed from it, and any cells that were not previously included in the output are added to it. • You can change the column widths and row heights of the grid by dragging the column and row header dividers. • If row range selection is supported, the row range you specify when creating the output value is displayed above the grid.
See also	"Define/Modify Row Range Dialog Box" on page 1467

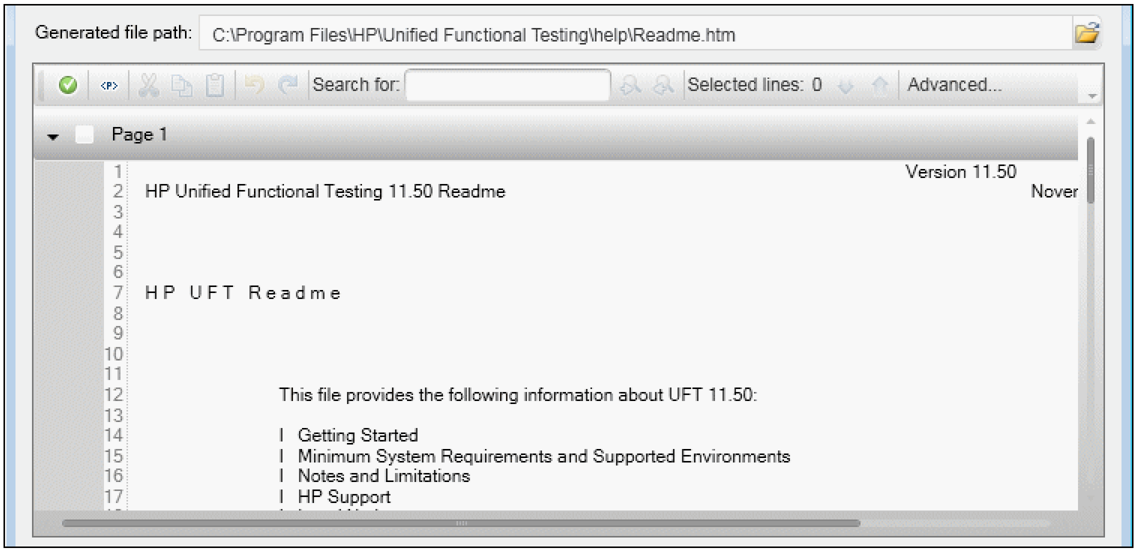
User interface elements are described below:

UI Element	Description
<grid area>	The grid area displays the captured\expected values of all the cells in the table. Only the ones with blue checkmarks are outputted. You can instruct UFT to output the entire table, specific rows, specific columns, or specific cells. **Note:** UFTchecks only cells containing a check mark.
Change	Enables you to define or modify the row range to output by opening the "Define/Modify Row Range Dialog Box" on page 1467.
☑ ⌗	**Add/Remove Output.** Add or remove the selected cells from the output.

File and Output Value Details Area (Output Value Properties Dialog Box) - File Content Output Values

Relevant for: GUI tests and scripted GUI components

When working with a file content output value, the Properties Grid area displays a File Content Editor which enables you to specify the text to output in a document that is generated or accessed during a run session.



Generated file path: C:\Program Files\HP\Unified Functional Testing\help\Readme.htm

```
                                                              Version 11.50
HP Unified Functional Testing 11.50 Readme                           Nover




H P  U F T  R e a d m e




          This file provides the following information about UFT 11.50:

          l  Getting Started
          l  Minimum System Requirements and Supported Environments
          l  Notes and Limitations
          l  HP Support
```

User interface elements are described below:

UI Element	Description
Generated file	The file path of the generated document from which you want to output text during the run session. You can enter a path manually or click the **Select Comparison File** button to navigate to the required file. The path can be relative or absolute (unless you use a regular expression). The file must be located in the file system. When you initially specify a file to compare, the textual content from that file is displayed in the file content editor of the dialog box as the expected value of the checkpoint. If you later change this path, the content in the file content editor is not affected. **Note:** The document must not be password-protected, otherwise, UFT will not be able to access it during a run session. You can use the following when specifying the document path: • **Configure Verification Value (Parameter/Regular Expression).** Opens the "Value Configuration Options Dialog Box" (described on page1579). You use this dialog box to define the file path as a constant or parameter value, with or without a regular expression. For example, suppose your application is being released in multiple languages, and you want to compare a different source file for each language. You can specify a data table parameter to instruct UFT to use a different file for each iteration. Displayed only when you hover over the **Comparison file path** text box. **Note:** If you specify a parameter value for the comparison file path and multiple files that match the value exist in the specified location, UFT performs the checkpoint on the latest generated file (according to its creation time). If you use a regular expression, you must specify a mapped drive, for example: C:\\Program Files\\HP\\Unified Functional Testing\\Tests\\.*\.txt • **Select Comparison File.** Enables you to navigate to the file you want to compare.
✓	**Select/Clear Line for Verification.** This toggle button enables you to select or clear a line to output in the generated file during a run session.

UI Element	Description
<P>	**Add Parameter to Line.** Opens the "Value Configuration Options Dialog Box". You use this dialog box to define a parameter as a constant or parameter value. For example, suppose you inserted a parameter in the "Generated file" box because you want to compare a different file during each iteration. You can specify a data table parameter to instruct UFT to use a different value for each iteration.
	Open Regular Expression Evaluator. Opens the "Value Configuration Options Dialog Box". You use this dialog box to create and test a regular expression enabling you to determine whether it suits your needs.
--	**Search for.** Text box in which you can optionally enter text for which to search. As you type, all instances of the specified text are highlighted. Press Enter to jump to the first instance if it is not visible in the file content editor. You can use the following buttons to navigate between each instance of the specified text: **Find Next.** Jumps to the next instance of the specified text in the file content editor. **Find Previous.** Jumps to the previous instance of the specified text in the file content editor. (Enabled only after you click **Find Next**.)
Advanced	Opens the "Advanced File Content Checkpoint Properties Dialog Box". This dialog box enables you to set additional comparison settings and checkpoint properties.
Page	The page from the source file being compared. Pages are displayed only when the content from the source file exceeds one page. You can: • View the number of pages in the file being compared. • Expand or contract a specific page to view or hide the content on that page. • Select the adjacent check box to select all of the lines on a page. • Clear the adjacent check box to clear the selection of all of the lines on a particular page.

Properties Grid Area (Output Value Properties Dialog Box) - Table Output Values

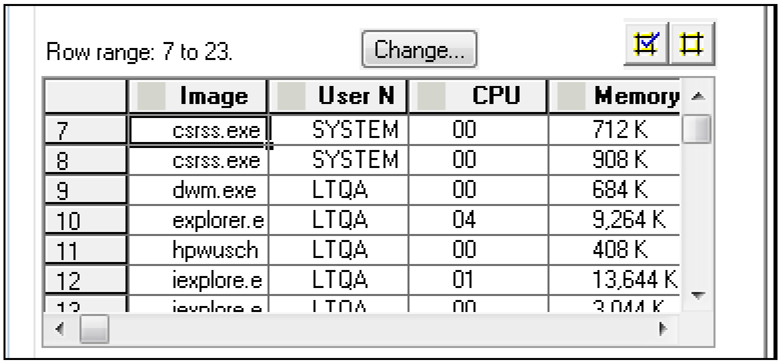
Relevant for: GUI tests and scripted GUI components

For a table output value, the Properties Grid area displays two tabs:

- "Grid Area" below
- "Properties Grid Area" on the next page

Grid Area

The following image shows the Grid area when row range selection is enabled.

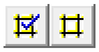


Row range: 7 to 23.

	Image	User N	CPU	Memory
7	csrss.exe	SYSTEM	00	712 K
8	csrss.exe	SYSTEM	00	908 K
9	dwm.exe	LTQA	00	684 K
10	explorer.e	LTQA	04	9,264 K
11	hpwusch	LTQA	00	408 K
12	iexplore.e	LTQA	01	13,644 K
13	iexplore.e	LTQA	00	3,044 K

Important information	- The column header names are captured from the table you selected for your output value. - Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the output are removed from it, and any cells that were not previously included in the output are added to it. - If row range selection is supported, the row range you specify when creating the output value is displayed above the grid.

User interface elements are described below:

UI Element	Description
<grid area>	The grid area displays the captured\expected values of all the cells in the table. Only the ones with blue checkmarks are outputted. You can instruct UFT to output the entire table, specific rows, specific columns, or specific cells. **Note:** UFT checks only cells containing a check mark.
Change	Enables you to define or modify the row range to output by opening the "Define/Modify Row Range Dialog Box" on page 1467.
	**Add/Remove Output.** Add or remove the selected cells from the output.

Properties Grid Area

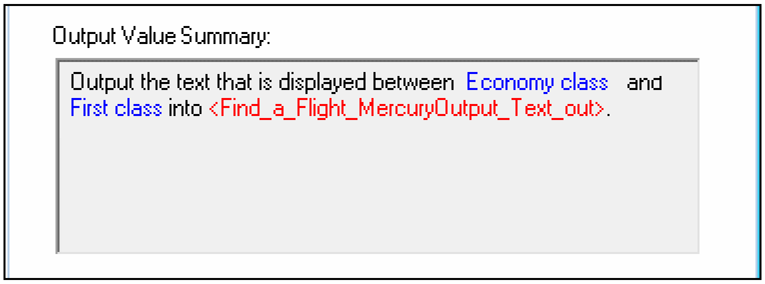
The Properties grid displays the table object's default properties, including the properties, their values, and their types. It is identical to the Properties Grid area of the Output Value Properties dialog box.

For user interface description, see "Properties Grid Area (Output Value Properties Dialog Box) - Standard Output Values" on page 1501.

Output Value Summary Area (Output Value Properties Dialog Box) - Text/Text Area Output Values

Relevant for: GUI tests and scripted GUI components

For a text or text area output value, the Properties Grid area displays a summary of the text to check.



Output Value Summary:

Output the text that is displayed between Economy class and First class into <Find_a_Flight_MercuryOutput_Text_out>.

User interface elements are described below:

UI Element	Description
Output Value Summary	Summarizes the selected text for the output value. It displays the text you selected when creating the output value, plus the text before and after it. UFT automatically displays the output text in red, and the text before and after the output text in blue. For text output values in Web-based environments, it displays the text you selected when creating the output value, plus some text before and after it. For text and text area output values in Windows-based environments, it displays the text you selected when creating the output value. **Note:** • In Windows-based environments, if there is more than one line of text selected, the Output Value Summary area displays **[complex value]** instead of the selected text string. You can then click **Configure** to view and manipulate the actual selected text for the output value. • For a text area output value, the output value string contains all the text in the selected area. Therefore, although the Text Output Value Properties and Text Area Output Value Properties dialog boxes are identical, when you create a text area output value, the **Text Before** and **Text After** values are not captured.

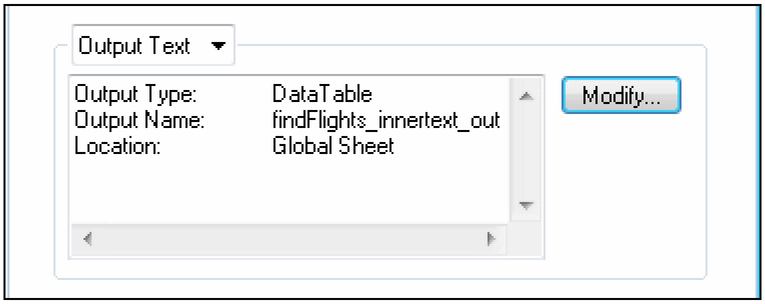
Output Text Options Area (Output Value Properties Dialog Box) - Text/Text Area Output Values

Relevant for: GUI tests and scripted GUI components

The Output Text Options area displays different options depending on whether you are checking the specific text to output, or the text before/after the output text:

- "Output Text Options Area " on the next page
- "Text Before / Text After Options" on the next page

Output Text Options Area



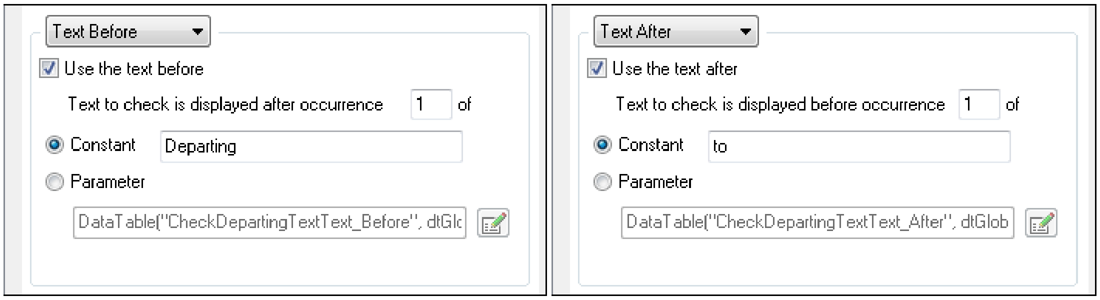
Output Text ▾

Output Type: DataTable
Output Name: findFlights_innertext_out
Location: Global Sheet

User interface elements are described below:

UI Element	Description
<Text Area list>	The current output value settings for the selected text. When you create a new output value, the default output definition is displayed for the value. For details, see "Default Output Definitions (tests only)" on page 1491.
Modify	Enables you to set parameterization and other preferences for each of the string elements in your output value, by opening the "Output Options Dialog Box " (described on page 1519).

Text Before / Text After Options



Text Before ▾

☑ Use the text before

Text to check is displayed after occurrence 1 of

● Constant Departing

○ Parameter

DataTable("CheckDepartingTextText_Before", dtGlc

Text After ▾

☑ Use the text after

Text to check is displayed before occurrence 1 of

● Constant to

○ Parameter

DataTable("CheckDepartingTextText_After", dtGlob

User interface elements are described below:

UI Element	Description
Use the text before / Use the text after	When selected, the current **Text Before / Text After** value is displayed in the **Constant** box. When cleared, UFT retrieves the value of the first occurrence of the defined output string, regardless of the text displayed before it (if you chose **Text Before**) or after it (if you chose **Text After**). **Note:** When this check box is cleared, the options below it are not available.

UI Element	Description
Text to capture is displayed before occurrence / **Text to capture is displayed after occurrence**	Specifies the exact occurrence of the value specified in the **Constant** or **Parameter** box, if the value is displayed more than once in the object or area. If you accept the default text that UFT recommends, the number in this box is correct. For example, if the selected output string is displayed before the first occurrence of the string First (as shown in the dialog box above). When **Text After** is selected, the number 1 is displayed in the **Text to capture is displayed before occurrence** box. If you modify the recommended value, you must confirm that the occurrence number is accurate. If you choose text that is not unique in the defined object or area, change the occurrence number appropriately. For example, if you want to output the text displayed after the third occurrence of the string Mercury Tours, select **Text Before** and enter 3 in the **Text to capture is displayed after occurrence** box. **Note:** UFTstarts counting occurrences of the specified **Text After** value from the beginning of the text string you selected to output, and includes any occurrences within the output value string itself.
Constant	Sets the **Text Before** or **Text After** value as a constant. A constant is a value that is defined directly within the test. It remains set for the duration of the test. When you are creating a text output value with **Text Before** selected, the **Constant** box displays the captured **Text Before** value. When you are creating a text output value with **Text After** selected, the **Constant** box displays the captured **Text After** value. You can change the value by typing in the text box. When you are creating a text area output value, the **Text Before** and **Text After** values are not captured. You can enter the text by typing or copying it into the **Constant** box. **Tip:** It is recommended to specify a text string that is unique within the object or area whenever possible, to ensure that the occurrence number is 1.
Parameter	Sets the **Text Before** or **Text After** value as a parameter. For details on specifying parameter values, see "How to Configure Constant and Parameter Values" on page 1572.

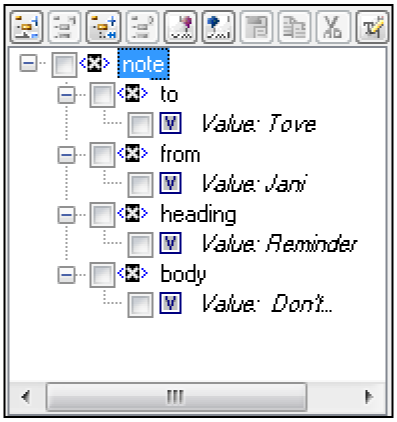
XML Tree/Options Area (Output Value Properties Dialog Box) - XML Output Values

Relevant for: GUI tests and scripted GUI components

For XML output values, the Output Value Properties dialog box displays an XML tree and options section to enable you to select the XML elements to output:

- "XML Tree" below
- "Options Area" on the next page

XML Tree



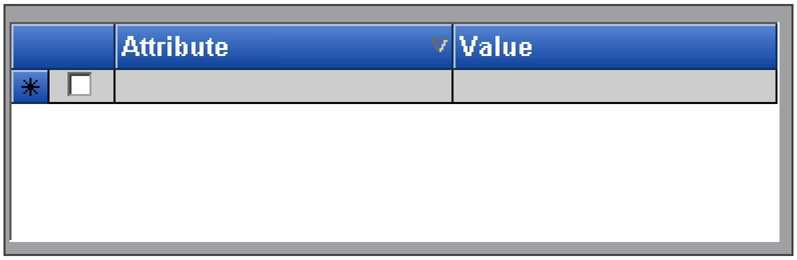
Important information	- The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the element and/or attribute values that you want to output. Each element node is displayed with a <⊠> icon. Each value node is displayed with a V icon. - When you create an XML output value for an operation return value, only a generic XML tree is created and shown in the XML Output Properties dialog box. Before you can select which element or attribute values you want to output, you must populate the XML tree with the actual elements, attributes, and values. - Select the check box for an element or value node in the XML tree to indicate that you want to output a value for that node. - Select an element node in the XML tree to display or set output options for its attributes and values on the right of the XML Output Properties dialog box.

The following commands are available according to the node you select in the tree:

Command	Icon	Description
Add Child		Adds a child node below the selected node in the tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the tree.
Add Value		Enables you to assign a constant or parameterized value to the selected element.

Command	Icon	Description
Delete		Deletes the selected node. Note that you cannot delete the root node of the output value step.
Import XML		Enables you to browse to and select a file structure from an existing XML file. The new file overrides the selected node's current sub-tree.
Export XML		Enables you to save the file structure of the selected node to an XML file.
Edit XML as Text		Opens the "Edit XML as Text Dialog Box", enabling you to modify the XML text of the selected node and it's subnodes in a test editor.
Duplicate		Adds a new node, identical to the selected one, as a sibling node at the same level as the selected node in the XML tree. **Note:** This command is available only from the context menu (right-click menu).

Options Area



User interface elements are described below:

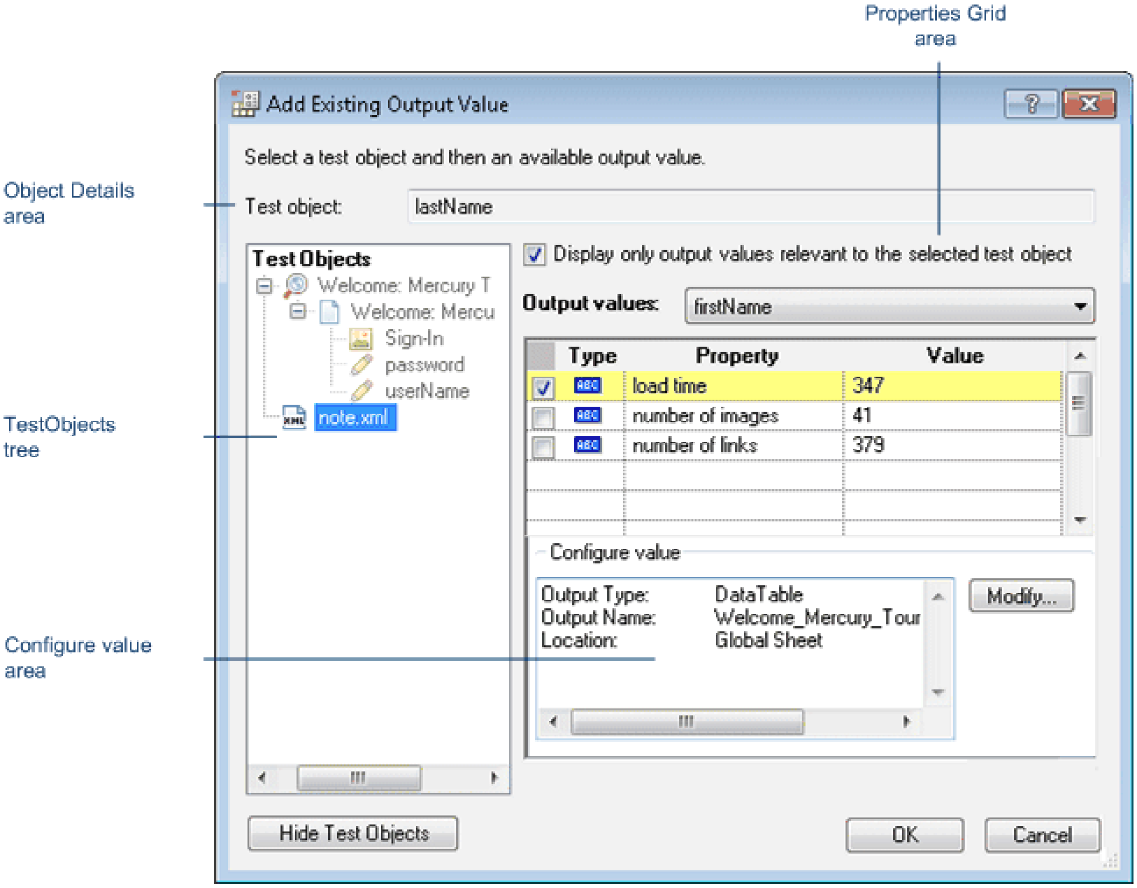
UI Element	Description
Attribute	The list of attributes for the selected element or value node in the XML tree.
Value	The list of values for the selected element or value node in the XML tree. **Note:** Click the **Configure the value** button to display the "Output Options Dialog Box ", which enables you to select or define the parameter in which you want to store the retrieved value. For details on the options available for each parameter type, see "Parameter Options Dialog Box" on page 1556.

Add Existing Output Value Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to select the test object to which you want to add an existing output value.

The following image shows an example of the Output Value Properties dialog box when the **TestObjects** tree is shown.



Select a test object and then an available output value.

Test object: lastName

☑ Display only output values relevant to the selected test object

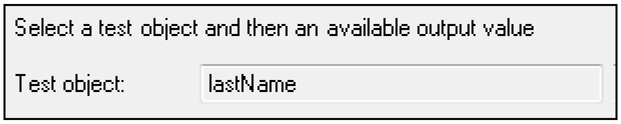
Output values: firstName

	Type	Property	Value
☑	ABC	load time	347
□	ABC	number of images	41
□	ABC	number of links	379

Output Type: DataTable
Output Name: Welcome_Mercury_Tour
Location: Global Sheet

To access	1. Ensure that a GUI action or component is in focus in the document pane. 2. Insert a new output value step and select the **Existing Output Value** option.
Important information	This option is available only if at least one of the object repositories associated with the current action (including the local object repository) contains at least one output object.
Relevant tasks	"How to Create or Modify an Output Value Step" on page 1494
See also	"Output Options Dialog Box " on page 1519

Object Details Area



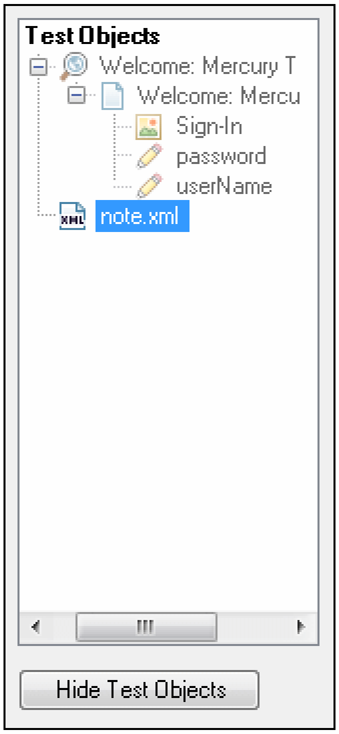
Select a test object and then an available output value

Test object: lastName

User interface elements are described below:

UI Element	Description
Test object	The test object for which you are adding an output value (read-only).

TestObjects Tree Area

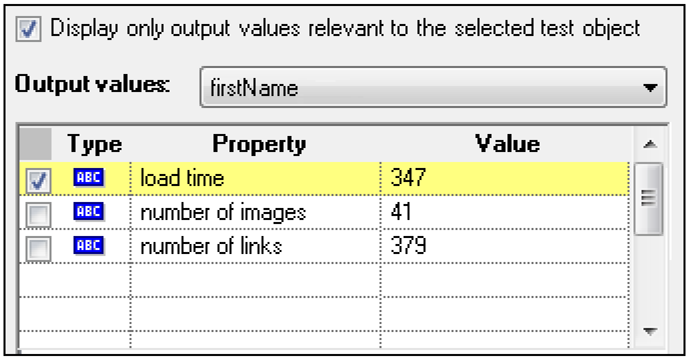


User interface elements are described below:

Option	Description
TestObjects tree	The objects stored in the object repositories associated with the current action.
Show/Hide Test Objects	Shows or hides the **TestObjects** tree.

Properties Grid Area

The following image shows the Properties Grid area when only part of the properties are selected. Specific properties may vary depending on the type of object for which you are defining output values.



☑ Display only output values relevant to the selected test object

Output values: firstName

	Type	Property	Value
☑	ABC	load time	347
□	ABC	number of images	41
□	ABC	number of links	379

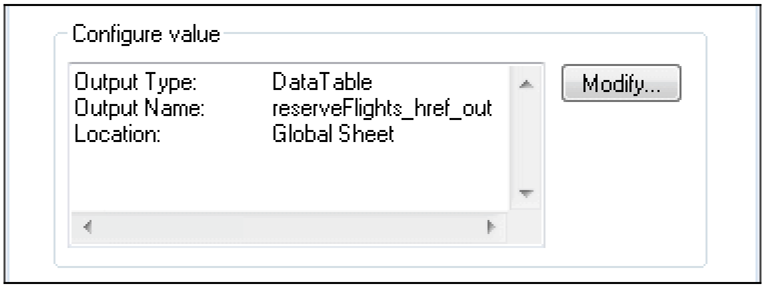
User interface elements are described below:

UI Element	Description
Display only output values relevant to the selected test object	Instructs UFT to determine which output value objects from the current action's object repositories are relevant for the selected object (based on the output value type and the properties selected to output in the output value object) and display only those output value objects in the Output Values list. **Note:** When using this option, it is recommended to open your application and display the selected object so that UFT can accurately determine all of the output values that can apply to that object.
Output values	The output values available for insertion. If the **Display only output values relevant to the selected test object** option is cleared, this list includes all output value objects from all object repositories associated with the current action. If the **Display only output values relevant to the selected test object** option is selected, this list displays only the relevant output value objects as described above.
Check box	Enables you to select one or more property for the object, and specify the output options for each property value you select.

UI Element	Description
Type	The ABC icon indicates that the value of the property is currently a constant. The icon indicates that the value of the property is currently a test or action parameter. The icon indicates that the value of the property is currently a DataTable parameter. The icon indicates that the value of the property is currently an environment variable parameter. The icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The expected value of the property.

Configure Value Area

The following image shows the Configure Value area when the Data pane is used to store the output value.



Output Type: DataTable
Output Name: reserveFlights_href_out
Location: Global Sheet

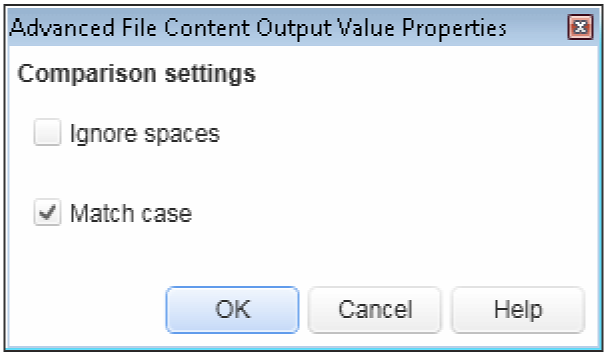
User interface elements are described below:

UI Element	Description
Configure value area	The output definition for the selected property, in read-only mode.
Modify	Opens the "Output Options Dialog Box " (described on page 1519), enabling you to change the output type and/or settings for the selected value.

Advanced File Content Output Value Properties Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to configure how the output value step is performed.



To access	In the "Output Value Properties Dialog Box" (described on page 1498), select **Advanced** in the toolbar.
Important information	When UFT performs a file content output value step, it locates the output value text within the line by comparing all of the text on that line. The options in this dialog box instruct UFT how to treat the text on every line in the file in which an output value is specified. (You can create multiple output values in a single file.)
Relevant tasks	"How to Create or Modify an Output Value Step" on page 1494
See also	"Output Value Properties Dialog Box" on page 1498

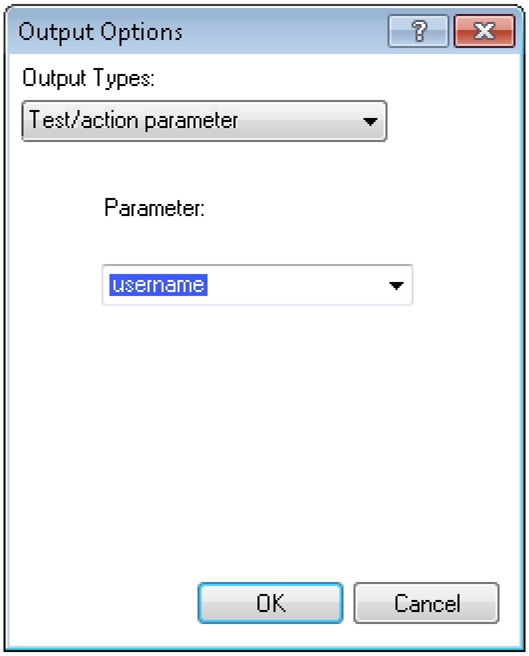
User interface elements are described below:

UI Element	Description
Ignore spaces	Ignores extraneous spaces on the same line as the output text when comparing the expected content with the actual content. This is useful when there might potentially be one or more spaces in the text preceding or following the checked text, and you do not want the checkpoint to fail because of this. **Note:** This setting is not relevant for missing spaces. **Default:** Cleared
Match case	Conducts a case-sensitive check of all the text on the same line as the output value text when comparing the expected content with the actual content. **Default:** Selected

Output Options Dialog Box

Relevant for: GUI tests and components

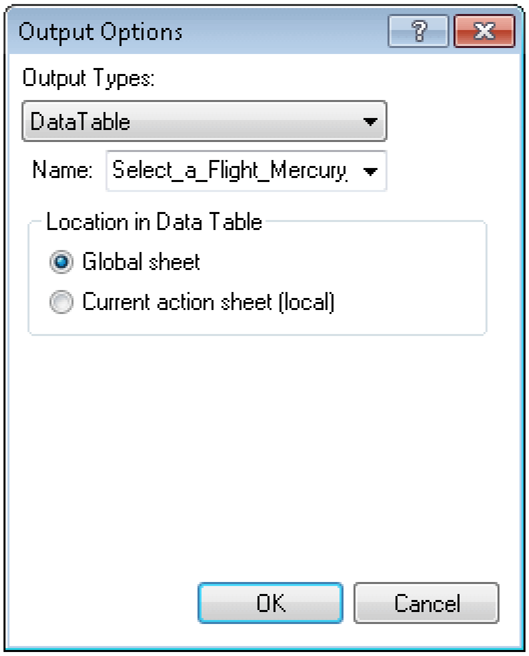
This dialog box enables you to define the output type for the specified property. The available output types depend on whether the output value is for a test, a scripted component, or a keyword component.



Output Options

Output Types: Test/action parameter

Parameter: username



Output Options

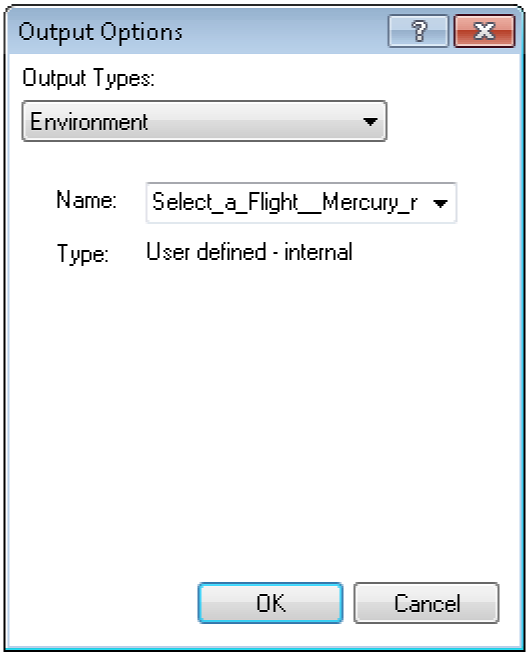
Output Types: DataTable

Name: Select_a_Flight_Mercury

Location in Data Table

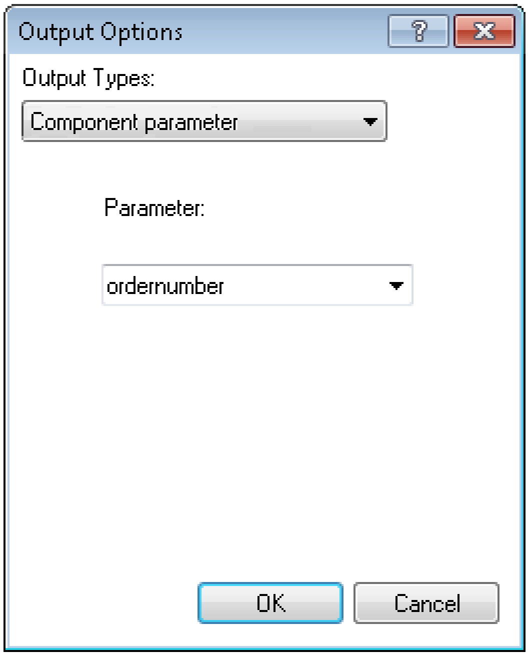
● Global sheet

○ Current action sheet (local)



Name: Select_a_Flight__Mercury_r

Type: User defined - internal



Parameter: ordernumber

To access	• **For tests and scripted components:** In the Configure Value area of any Output Value Properties dialog box, click the **Modify** button. • For **keyword components**, use one of the following:. ▪ In the **Output** cell of a component step, click the output value button . ▪ Press **CTRL + F11.**
Important information	• You can only output a value to an action parameter if the parameter has been defined as an output parameter for the calling action. • If at least one output parameter is defined in the action, UFT may display **Test/action parameter** as the default output type. Otherwise, **DataTable** may be displayed as the default output type. • If at least one output component parameter is defined in the component, the default output type is Component parameter and the default output name is the first output parameter displayed in the Parameters pane of the Business Component Settings dialog box. • After you define an output parameter for a component, you can use its value as an input value in a later step in the component (local and component parameters), or in a later component in the business process test (component parameters only). • If you click in the **Output** cell in a component step after you specify an output parameter for the component, an icon specifying the type of parameter is displayed in the cell: ▪ Indicates a component parameter. ▪ Indicates a local parameter.
See also	• "Output Value Properties Dialog Box" on page 1498

You can choose from the following output types:

- "Test/Action Parameter Option" on the next page
- "Data Table Option" on the next page
- "Environment Variable Option" on page 1524
- "Component Parameter Option" on page 1524

Test/Action Parameter Option

UI Elements	Description
Test/action parameter Output type	Enables you to output a value to an action parameter, so that the values can be used later in the run session, or the values can be passed back to the external application that ran (called) the test. For details on outputting a value to a test or action parameter, see "Storing Output Values". For details on creating and using parameters, see "Parameterizing Object Values" on page 1526.
Parameter	The name of the parameter in which to store the output value. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters for the action. **Tip:** You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

Data Table Option

UI Elements	Description
Data Table Output type	Enables you to output values to the run-time data table. This enables you to store and retrieve values during a data-driven test (or action) that runs several times. In each repetition, or **iteration**, UFT stores the value in a different row within the Data pane.
Name	The name of the column in the Data pane in which to store the value. UFT suggests a default name for the output. You can select an existing output name from the list, or create a new output name by using the default output name or entering a valid descriptive name.
Location in Data Table	Specifies whether to add the DataTable parameter (column name) in the global or current action sheet in the data table. For details on the use of data in the global and current action sheets, see "Action and Test Iterations Using the Data Pane" on page 874. For details on actions, see "Actions in GUI Testing" on page 871. For details on outputting values to the Data pane, see "Run-time Data Table".

Environment Variable Option

UI Elements	Description
Environment Output type	Enables you to specify the internal user-defined environment variable in which to store the selected value for the duration of the run session.
Name	The name of the internal user-defined environment variable in which to store the value. The list contains all currently defined internal user-defined environment variables with the corresponding type. You can select an existing variable from the list, or you can create a new internal environment variable by modifying the displayed name or by entering a new, descriptive name. **Note:** If you select an existing variable from the list, UFT prompts you to choose whether to overwrite its current value with the new value when the output value step runs. If you choose not to overwrite the current value of the selected variable, a new environment variable is created with the original variable name and an identifying suffix.
Type	The environment variable type. Because it is not possible to output values to external or built-in environment variables, the type is always **User-defined - internal**. For details on environment variables, see "Environment Variable Parameters" on page 1536.

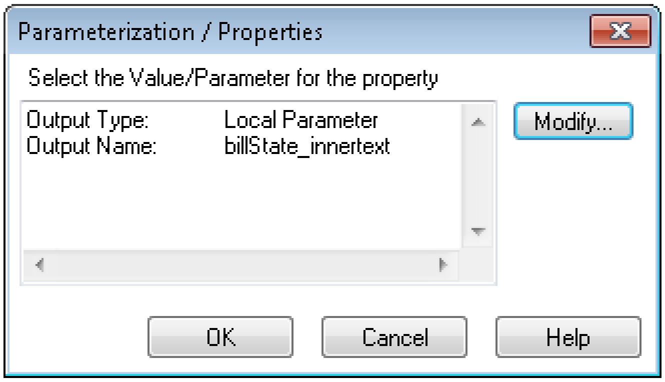
Component Parameter Option

UI Elements	Description
Output Types	The type of output parameter. Possible values: • **Component parameter** • **Local parameter**
Parameter	The component parameter in which to store the output value. (Displayed only if **Component parameter** is **Output Type**.) The names and full descriptions of the available component parameters are displayed as read-only. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.
Name	The name of the parameter. (Displayed only if **Local parameter** is the **Output Type**) Enter a meaningful name for the parameter or choose one from the list. If no local parameter is defined, then the default parameter name, **p_Local**, is displayed.
Description	A brief description for the parameter. (Displayed only if **Local parameter** is **Output Type**.)

Parameterization / Properties Dialog Box (Output Values)

Relevant for: Keyword GUI components

This dialog box displays output property value as a **Constant** or a **Parameter**, in read-only mode.


Parameterization / Properties
Select the Value/Parameter for the property
Modify...
OK
Cancel
Help

Output Type: Local Parameter
Output Name: billState_innertext

To access	In the Output Value Properties dialog box (Simple mode), click the **Browse** button for a property ... .
Important information	- This option is available only in Simple mode. When in Advanced mode, the options in this dialog box are available in the **Configure value** area of the Output Value Properties dialog box. - You edit the output definition for output values in the Output Options dialog box. For details, see "Component Parameter Option" on the previous page.
Relevant tasks	"How to Create or Modify an Output Value Step" on page 1494
See also	- "Output Value Properties Dialog Box" on page 1498 - "Output Values Overview" on page 1488

For a description of user interface elements, see "Configure Value Area (Checkpoint Properties Dialog Box) - Standard/Image Checkpoints" on page 1436.

Chapter 52: Parameterizing Object Values

Relevant for: GUI tests and scripted GUI components

This chapter includes:

Concepts

Parameterizing Values Overview

Relevant for: GUI tests only

You can enhance your test by parameterizing the values that it uses. A **parameter** is a variable that is assigned a value from an external data source or generator.

You can parameterize values of:

- Checkpoints.
- Object properties for a selected step.
- Operation arguments defined for a selected step.
- One or more properties of an object stored in the local object or the "Object Repository Window" (described on page 1274). For details on parameterizing a property value for an object in a shared object repository, see "Shared Object Repositories" on page 1285.

Example

> Your application may include a form with an edit box into which the user types the user name. You may want to test whether your application reads this information and displays it correctly in a dialog box. You can insert a text checkpoint that uses the built-in environment variable for the logged-in user name, to check whether the displayed information is correct.
>
> **Note:** When you parameterize the value of an object property for a local object, you are modifying the test object description in the local object repository. Therefore, all occurrences of the specified object within the action are parameterized. For details on the local object repository, see "Managing Test Objects in Object Repositories" on page 1198.

You can parameterize the values in steps or the values of action parameters using one of the following parameter types:

- **Test/action parameters.** Test parameters enable you to use values passed from your test. Action parameters enable you to pass values from other actions in your test. For details, see "Test and Action Input Parameters" on the next page.
- **DataTable parameters.** Enable you to create a data-driven test (or action) that runs several times using the data you supply. In each repetition, or iteration, UFT uses a different value from the Data pane. For details, see "Data Table Parameters" on page 1533.
- **Environment variable parameters.** Enable you to use variable values from other sources during the run session. These may be values you supply, or values that UFT generates for you

based on conditions and options you choose. For details, see "Environment Variable Parameters" on page 1536.

- **Random number parameters.** Enable you to insert random numbers as values in your test. For example, to check how your application handles small and large ticket orders, you can instruct UFT to generate a random number and insert it in a **number of tickets** edit box.

Tip:

- If you want to parameterize all the operation arguments in your test or in one or more actions of a test, consider using the Automatically parameterize steps option. For details, see "Automatically Parameterizing Steps" on page 1539.
- If you want to parameterize the same value in several steps in your test, consider using the Data Driver rather than adding parameters manually. For details, see "Data Driver" on page 1543.
- When you use the Step Generator to add new steps, you can parameterize the values for the operation you select. For details, see "How to Insert Steps Using the Step Generator" on page 978.
- You can also parameterize identification property values of test objects in the object repository using repository parameters. For details, see "Working with Repository Parameters" on page 1287.

Test and Action Input Parameters

Relevant for: GUI tests only

You can parameterize a value in a step using a test or action input parameter. To use a value within a specific action, you must pass the value down through the action hierarchy of your test to the required action. You can then use that parameter value to parameterize a step in your test.

You can parameterize a value using a test or action parameter only if the parameter has been defined for the test or action. For details on defining parameters, see "Parameters Tab (Properties Pane - GUI Testing)" on page 394, "Parameters Tab (Action Properties Dialog Box)" on page 903, and "Action Call Properties Dialog Box" on page 895.

Example

Suppose that Action3 is a nested action of Action1 (a top-level action), and you want to parameterize a value in Action3 using a value that is passed into your test from the external application that runs (calls) the test.

You can pass the value from the test level to Action1, then to Action3, and then parameterize the required value using this action input parameter value (that was passed through from the external application).

Alternatively, you can pass an output action parameter value from an action step to a later sibling action at the same hierarchical level. For example, suppose that Action2, Action3, and Action4 are sibling actions at the same hierarchical level, and that these are all nested actions of Action1.

You can parameterize a call to Action4 based on an output value retrieved from Action2 or Action3. You can then use these parameters in your action step.

For details, see "Considerations for Setting Action Parameters" on the next page.

This section also includes:

- "How to Parameterize a Value" below
- "Using Action Parameters in Steps in the Editor" below
- "Considerations for Setting Action Parameters" on the next page

How to Parameterize a Value

You parameterize values by selecting input parameters in the Parameter Options or "Value Configuration Options Dialog Box" (described on page 1579). The parameter options that are available in these dialog boxes depend on where you are currently located in your test, and whether test or action parameters are defined.

Alternatively, you can enter the parameter name in the Editor using the **Parameter** utility object. For details, see "Using Action Parameters in Steps in the Editor" below.

For details, see "How to Parameterize Values for Operations or Local Objects" on page 1549 and "How to Parameterize a Checkpoint Property Value" on page 1550.

Tip: You can also create test or action parameter output values that retrieve values during the run session and store them for use at another point in the run session. You can then use these output values to parameterize a step in your test. For details, see "Storing Output Values" on page 1492.

For details, see "Parameter Options Dialog Box" on page 1556.

Using Action Parameters in Steps in the Editor

Instead of selecting input (or output) parameters from the appropriate dialog boxes while parameterizing steps or inserting output value steps, you can enter input and output parameters as values in the Editor using the **Parameter** utility object in the format: Parameter("ParameterName") for the current action, or Parameter("ActionName", "ParameterName") to use the output parameter from a previous action as an input parameter in the current action.

Example:

Suppose you have test steps that enter information in a form to display a list of purchase orders in a table, and then return the total value of the orders displayed in the table.

You can define input parameters, called **SoldToCode** and **MaterialCode**, for the codes entered in the **Sold to** and **Materials** edit boxes of the form so that the Orders table that is opened is controlled by the input parameter values passed when the test is called.

You can define an output parameter, for example **TotalValue**, to store the returned value. The output value (**TotalValue**) can then be returned to the application that called the test.

The example described above might look something like this (parameters are in bold font):

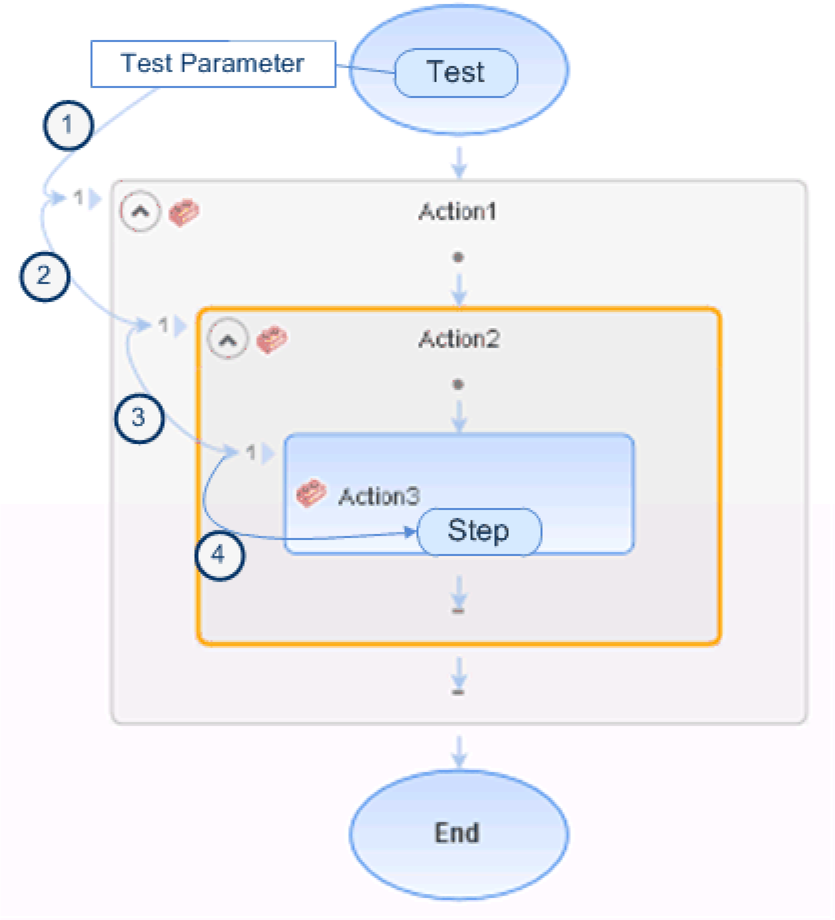
```
Browser("Mercury").Page("List Of Sales").WebEdit("Sold to").Set Parameter("SoldToCode")

Browser("Mercury").Page("List Of Sales").WebEdit("Materials").Set Parameter("MaterialCode")

Browser("Mercury").Page("List Of Sales").WebButton("Enter").Click

NumTableRows = Browser("Mercury").Page("List Of Sales").WebTable("Orders").RowCount

Parameter("TotalValue") = Browser("Mercury").Page("List Of Sales").WebTable("Orders").GetCellData(NumTableRows,"Total")
```

Considerations for Setting Action Parameters

- Input action parameter values can be used only within the steps of the current action. You can use an action input value from another action (or from the test) only if you pass the value from action to action down the test hierarchy to the action in which you want to use it.

- In subsequent steps of a calling action, you can use any type of action output value as a variable, if the value was retrieved from the called action. For example, if ActionA calls ActionB and specifies MyBVar as the variable in which to store ActionB's output parameter, then steps in ActionA after the call to ActionB can use the MyBVar as a value (just as you would use any other variable). For example:

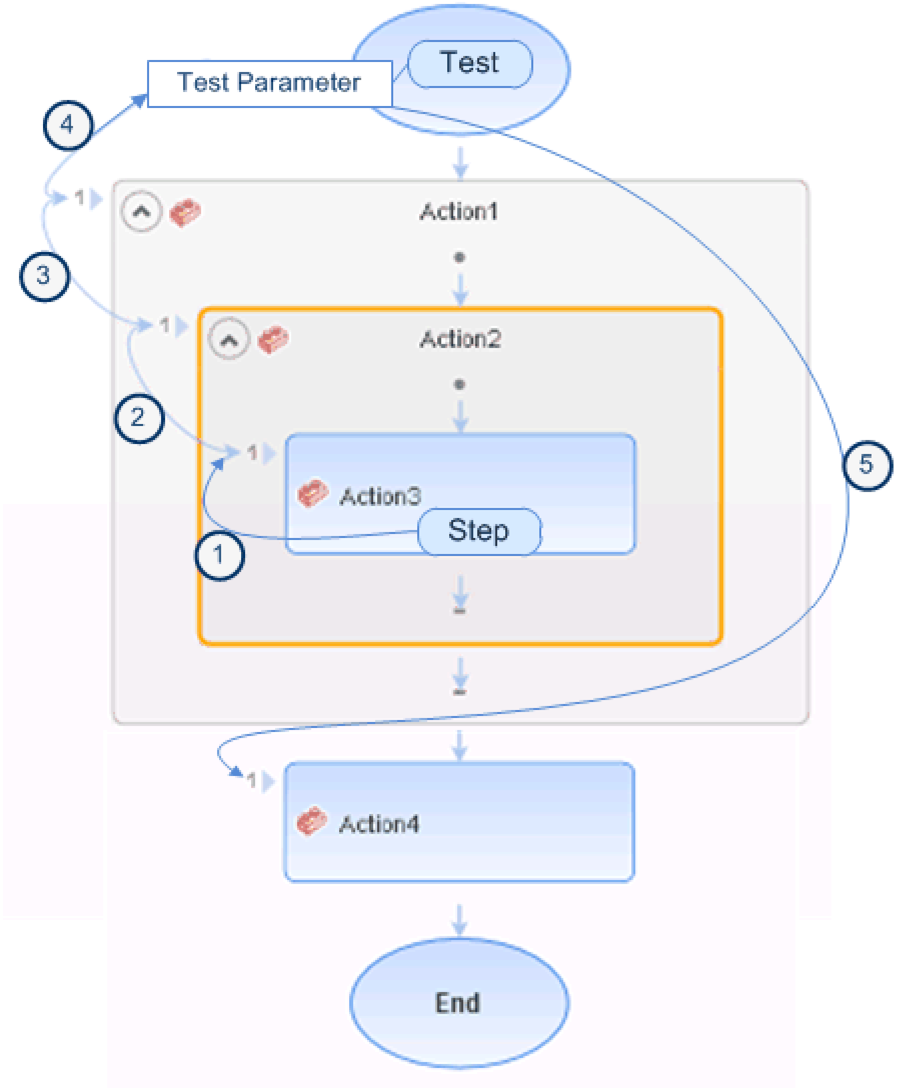
  Test -> Action1 -> Action2 -> Action3 -> (Action3) Step 1



- Output action parameter values can be retrieved from a previous action at the same hierarchical level, from a parent action, or from the current action. You can use an action output value from one action within the step of another action if:

  - You pass the value from action to action up the test hierarchy to the action in which you want to use it. For example:

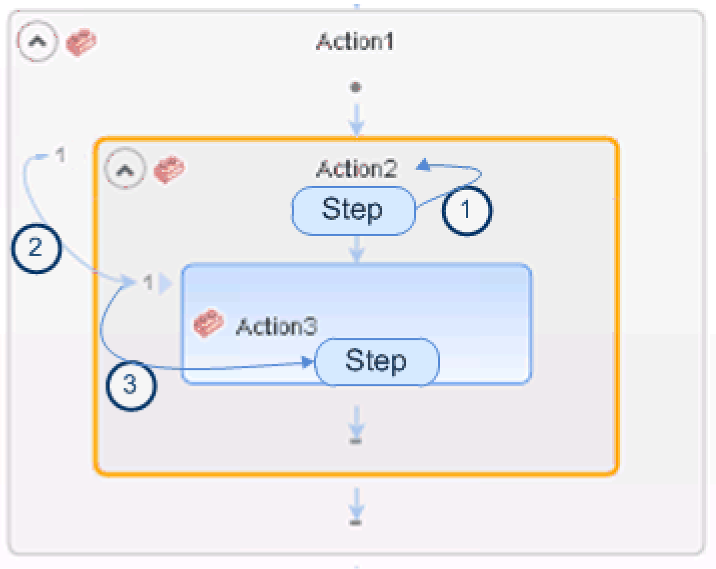
(Action3) Step 1-> Action3 -> Action2 -> Action1 -> Test -> Action4



In this example, any step in Action 1, Action 2, or Action 3 can potentially use the output value from (Action3) Step 1, even though the example shows that the output value is used by steps in Action4.

- You pass the value from a previous action to the sibling action in which you want to use it. For example:

  (Action2) Step 1 -> Action2 -> Action3 -> (Action3) Step 1

  In this example, any step in Action 2 or Action 3 can potentially use the output value from (Action2) Step 1, even though the example shows that the output value is used by (Action3) Step 1.

- You can link action parameters with test parameters using the Parameter Values tab in the Action Call Properties dialog box (described on page 897).

Data Table Parameters

Relevant for: GUI tests and scripted GUI components

You can supply the list of possible values for a parameter by creating a data table parameter. **Data table parameters** enable you to create a data-driven test, or action that runs several times using the data you supply. In each repetition, or **iteration**, UFT uses a different value from the Data pane (taken from the subsequent row in the Data pane).

When you create a new data table parameter, a new column is added at the end of the Data pane and the current value you parameterized is placed in the first row. If you parameterize a value and select an existing data table parameter, the values in the column for the selected parameter are retained and are not overwritten by the current value of the parameter.

> **Note:** If you parameterize a value that is defined as a variant value, then when UFT retrieves the value from the Data pane, it retrieves it as a string. This occurs even if you enter a numeric value in the Data pane. For example, if you parameterize the argument of a step such as:
> WpfWindow("MyWindow").WpfComboBox("cb").Select 1
> and you enter the value 1 in the Data pane, then when the step runs, it retrieves the value as a

string: "1", and the step fails.

A data table comprises:

- **Columns.** Each column in the table represents the list of values for a single data table parameter.
- **Column headers.** The data table parameter names.
- **Rows.** Each row in the table represents a set of values that UFT submits for all the parameters during a single iteration of the test. When you run your test, UFT runs one iteration of the test for each row of data in the Data pane. For example, a test with ten rows in the Global sheet of the Data pane runs ten times.
- **Sheets.** The Data pane contains two types of sheets—Global and action-specific. For details, see "Data Table Sheets for GUI Testing" on page 225.

For details on entering values in the Data pane, see "Data Pane" on page 221.

Tip: You can also create data table output values, which retrieve values during the run session and insert them into a column in the Data pane. You can then use these columns as data table parameters in your test. For details, see "Output Values Overview" on page 1488.

Example 1

Suppose your application includes a feature that enables users to search for contact information from a membership database. When the user enters a member's name, the member's contact information is displayed, together with a button labelled **View <MemName>'s Picture**, where **<MemName>** is the name of the member. You can parameterize the name property of the button using a list of values so that during each iteration of the run session, UFT can identify the different picture buttons.

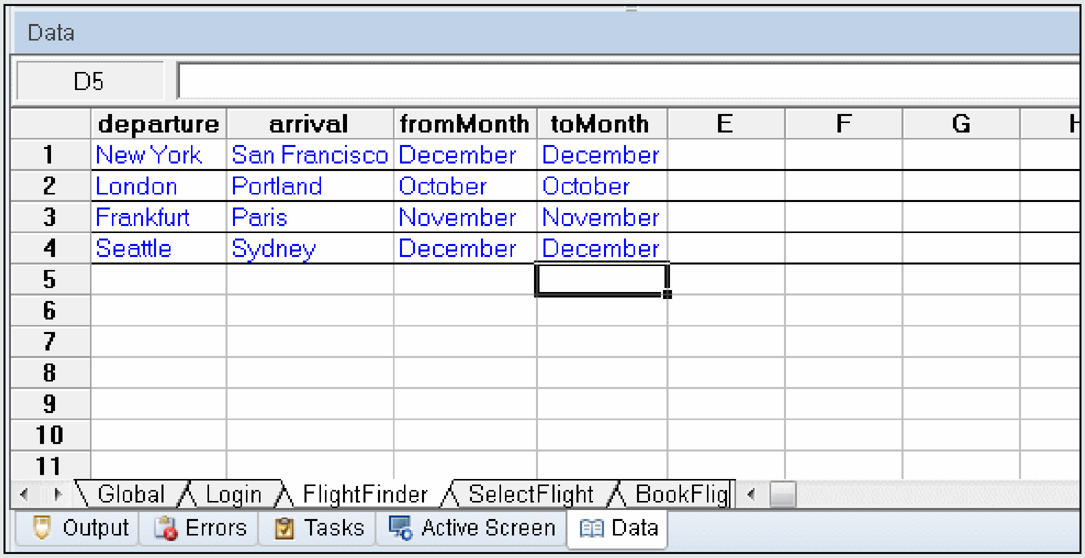
Example 2

Consider the Mercury Tours sample Web site, which enables you to book flight requests. To book a flight, you supply the flight itinerary and click the **Continue** button. The site returns the available flights for the requested itinerary.

Although you could conduct the test by accessing the Web site and submitting numerous queries, this is a slow, laborious, and inefficient solution. By using data table parameters, you can run the test for multiple queries in succession.

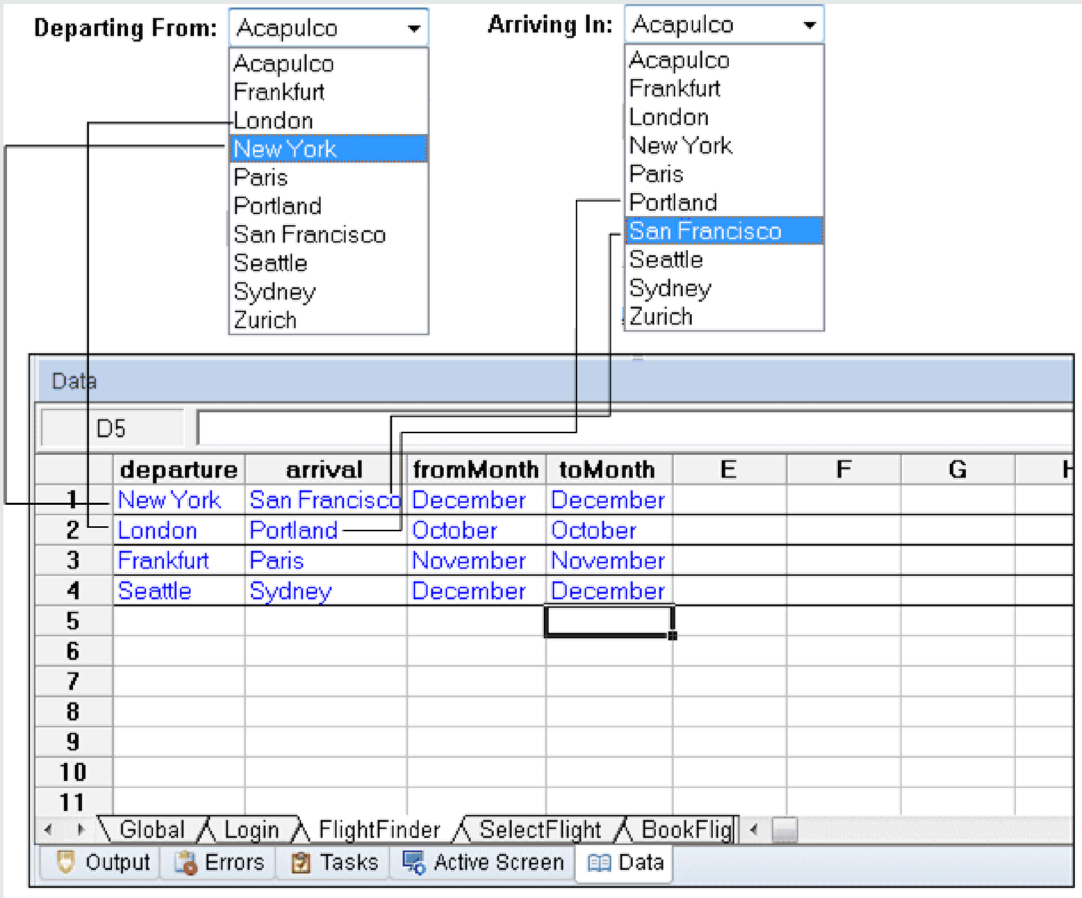
When you parameterize your test, you first create steps that access the Web site and check for the available flights for one requested itinerary.

You then substitute the existing itinerary with a data table parameter and add your own sets of data to the relevant sheet of the Data pane, one for each itinerary.



	departure	arrival	fromMonth	toMonth	E	F	G	H
1	New York	San Francisco	December	December				
2	London	Portland	October	October				
3	Frankfurt	Paris	November	November				
4	Seattle	Sydney	December	December				
5								
6								
7								
8								
9								
10								
11								

In this example, UFT submits a separate query for each itinerary when you run the test.



	departure	arrival	fromMonth	toMonth	E	F	G	H
1	New York	San Francisco	December	December				
2	London	Portland	October	October				
3	Frankfurt	Paris	November	November				
4	Seattle	Sydney	December	December				
5								
6								
7								
8								
9								
10								
11								

Environment Variable Parameters

Relevant for: GUI tests and scripted GUI components

UFT can insert a value from the Environment variable list, which is a list of variables and corresponding values that can be accessed from your test or component. Throughout the run session, the value of an environment variable remains the same, regardless of the number of iterations, unless you change the value of the variable programmatically in your test.

Example

You can instruct UFT to read all the values for filling in a Web form from an external file, or you can use one of UFT's built-in environment variables to insert current information about the computer running the test or scripted component.

Tip: Environment parameters are especially useful for localization testing, when you want to test an application where the user interface strings change according to the selected language. Environment parameters can be used for testing the same application on different browsers. You can also vary the input values for each language by selecting a different data table file each time you run the test. For details, see "Data Pane" on page 221.

There are several types of environment variables:

User-Defined Internal Environment Variables (tests only)

User-defined internal environment variables are defined within the test. These variables are saved with the test and are accessible only within the test in which they were defined.

You can create or modify internal, user-defined environment variables for your test in the:

- Environment pane of the Test Settings dialog box, as described in "Environment Pane (Test Settings Dialog Box)" on page 608.

- Parameter Options dialog box, as described in "Parameter Options Dialog Box" on page 1556.

Tip: You can also create environment output values, which retrieve values during the test run and output them to internal environment variable parameters for use in your test. For details, see "Output Values Overview" on page 1488.

User-Defined External Environment Variables (tests only)

User-defined external environment variables are predefined in the active external environment variables file. You can create as many files as you want and select an appropriate file for each test, or change files for each test run. External environment variable values are designated as read-only within the test.

External environment variable files are comprised of a list of variable-value pairs in .xml format. You select the active external environment variable file for a test in the Environment pane of the Test

Settings dialog box (see "Environment Pane (Test Settings Dialog Box)" on page 608). Then you can use the variables from the file as parameters.

You can set up your environment variable XML files manually, or you can define the variables as internal environment variables in the Environment pane of the Test Settings dialog box and use the **Export** button to create the .xml file with the correct structure.

For details on creating and using user-defined external environment variable files see, "How to Use User-Defined External Environment Variables" on page 1551.

Note:

- You can also store environment variable files in ALM. For details, see "Environment Variable Files and ALM (tests only)" below.
- You can create several external variable files with the same variable names and different values and then run the test several times, using a different file each time. This is especially useful for localization testing.

Built-in Environment Variables

Variables that represent information about the test or scripted component and the computer on which they are run, such as Test path and Operating system. These variables are accessible from all tests and scripted components, and are designated as read-only. For details, see "Environment Pane (Test Settings Dialog Box)" on page 608.

ALM provides a set of built-in variables that enable you to use current information about the test or scripted component and the UFT computer running your test. These can include the test or component name and path, the operating system type and version, and the local host name.

For example, you may want to perform different checks in your test or component based on the operating system being used by the computer that is running the test. To do this, you could include the OSVersion built-in environment variable in an If statement.

You can also select built-in environment variables when parameterizing values. For details, see "Parameter Options Dialog Box" on page 1556.

Note: UFT also has a set of predefined environment variables that you can use to set the values of the Record and Run Settings dialog options. You should not use the names of these variables for any other purpose. For details, see the sections on how to define record and run settings for Windows-based and Web-based applications in the *HP Unified Functional Testing Add-ins Guide*.

Environment Variable Files and ALM (tests only)

When working with ALM and environment variable files, you must save the environment variable file in the Test Resources module in your ALM project before you specify the file in the Environment pane of the Test Settings dialog box.

You can add a new or an existing environment variable file to your ALM project. Adding an existing file from the file system to an ALM project creates a copy of the file in ALM. Therefore, changes made to the ALM environment variable file do not affect the file system file and vice versa.

When to Choose Global or Action Data Table Parameters

Relevant for: GUI tests only

When you parameterize a step in a test using the Data pane, you must decide whether you want to make it a **global data table parameter** or a **local data table parameter**.

You use local data table parameters when you want the data to be used only for a single action. You use global data table parameters when you want the data to be available to other actions, and when you want subsequent iterations to use different data for a particular parameter (each time the test repeats or each time the action repeats within the test).

If you have multiple rows in the Global data sheet, the entire test runs multiple times. If you have multiple rows in a local data sheet, the corresponding action runs multiple times before running the next action in the test. If you have multiple rows in both Global and local data sheets, each single test iteration runs all iterations of each action before running the next iteration of the test.

- **Global data table parameters** take data from the Global sheet in the Data pane. The Global sheet contains the data that replaces global parameters in each iteration of the test.

  By default, the test runs one iteration for each row in the Global sheet of the Data pane. You can also set the test to run only one iteration. You can also set the test to run iterations on specified rows within the Global sheet of the Data pane.

  You can use the parameters defined in the Global data sheet in any action.

  For details on setting global iteration preferences, see "Run Pane (Test Settings Dialog Box)" on page 601.

  > **Tip:** By outputting values to the global Data pane sheet from one action and using them as input parameters in another action, you can pass values from one action to another. For details, see "Output Values Overview" on page 1488.

- **Local data table parameters** take data from the action's sheet in the Data pane. The data in the action's sheet replaces the action's data table parameters in each iteration of the action. By default, actions run only one iteration.

  You can also set a particular call of the action to run iterations for all rows in the action's sheet or to run iterations on specified rows within the action's sheet. When you set your action properties to run iterations on all rows, UFT inserts the next value from the action's data sheet into the corresponding action parameter during each **action iteration**, while the values of the global parameters stay constant. For details on setting action iteration preferences, see "Run Tab (Action Call Properties Dialog Box)" on page 895.

After running a parameterized test, you can view the actual values taken from the Data pane in the run results run-time data table. For details, see the section on the Run Results Viewer Data Pane (described in the *HP Run Results Viewer User Guide*).

Automatically Parameterizing Steps

Relevant for: GUI tests only

You can instruct UFT to automatically parameterize the steps in your test actions at the end of a recording session.

This enables you to create actions that can be used for a variety of different purposes or scenarios by referencing different sets of data.

To activate this option, select the **Automatically parameterize steps** option in the **General** node of the GUI Testing tab of the Options dialog box (**Tools > Options > GUI Testing** tab > **General** node). You can set the option to use **Global Data Table Parameters** or **Test Parameters**. For details, see "General Pane (Options Dialog Box > GUI Testing Tab)" on page 536.

When you stop a recording session while this option is selected, UFT replaces the constant values in the test object operation arguments of your steps with either Data pane parameters or action parameters, based on your selection of global Data pane parameters or test parameters in the Options dialog box.

UFT performs this automatic parameterization for all relevant steps in any action in your test, in which you recorded one or more steps during that recording session.

- When you select to use **Global Data Table Parameters**, the generated parameters are added to the Global sheet of the Data pane.

  If you work with ALM, you can map these parameters to the column names of a data resource and then use different configurations in your test sets. For details on ALM configurations and mapping data table parameters to ALM data resources, see "Data Awareness in ALM" on page 715.

- When you select to use **Test Parameters**, UFT parameterizes the step with a newly created action parameter. It also creates a corresponding test parameter.

For details on how UFT creates these parameters, which types of operation arguments are included and excluded from the parameterization, and other important criteria to consider, see "Guidelines and Considerations for Automatically Parameterizing Steps" on the next page.

This section also includes:

Guidelines and Considerations for Automatically Parameterizing Steps

Relevant for: GUI tests only

Before you begin a recording session with the **Automatically parameterize steps** option activated, make sure you are aware of the following guidelines and considerations:

General Guidelines

- The automatic parameterization process runs on all relevant steps in all local actions in which you recorded one or more steps during the recording session. It also parameterizes the action that is displayed when you stop the recording session, even if you did not record any steps in that action.
  - All relevant steps in any relevant action are parameterized, whether or not those steps were added in the current recording session. Therefore, you can perform automatic parameterization on an existing action by starting and stopping a recording session without adding any steps.
  - Automatic parameterization is not performed on external actions that are called from your test, nor for actions called using the **LoadAndRunAction** statement.
- In general, simple, constant (string, number, boolean) test object and utility object operation arguments are parameterized. Therefore, if the following are contained in a method argument, they are not parameterized:
  - arguments that are already parameterized
  - variables
  - enumeration constants, such as **micLeftbtn**
  - expressions, such as: x = 3
  - Assignments of values, such as: Window("Notepad").WinMenu("Menu").ExpandMenu = True
  - mathematical or other concatenation operations, such as "Hello World" & micCtrl & "S"
  - VBScript statements, such as msgbox "hello"
  - VBScript language statements such as For, If, Do, While
  - steps inside called functions from function libraries or in functions or sub-routines defined directly within the action

- In addition to the above general rule, operation arguments in the following scenarios are also not parameterized:
    - **SAPGuiTable.Input**, **SAPGuiGrid.Input**, or **SAPGuiAPOGrid.Input** steps (inserted using the **Auto-parameterize table and grid controls** option). For details, see the auto-parameterize table topic in the SAP section of the *HP Unified Functional Testing Add-ins Guide*.
    - Native methods and properties accessed by the **Object** property.
    - Steps for test objects that are stored in variables. For example, the "text" constant is not parameterized in:

      ```
      Set MyEditBox = Browser("x").Page("x").WebEdit("myedit")
      MyEditBox.Set "text"
      ```

    - Steps containing programmatic descriptions, such as:

      ```
      Browser("x").Page("x").WebEdit(MyDescription).Set "text"
      ```

      or

      ```
      Browser("x").Page("x").WebEdit("prop:=value", "prop2:=value2).Set "text".)
      ```

    - User-defined functions that are registered to test objects using a **RegisterUserFunc** statement.

      However, built-in UFT test object operations that were overridden using a **RegisterUserFunc** statement are parameterized.

    - Operation arguments of type variant, when the value is a number.

      For example, in the following statement, the variant argument of the Select method is not parameterized, because it is a number.

      ```
      WpfWindow("MyWindow").WpfComboBox("cb").Select 1
      ```

      However, in the following statement, the variant argument of the Select method is parameterized, because it is a string.

      ```
      WpfWindow("MyWindow").WpfComboBox("cb").Select "item1"
      ```

    - When working with Siebel applications, only operation arguments of **Sbl*** test objects, which represent standard interactivity (SI) objects are parameterized. The operation arguments of

Sieb* test objects, which represent Siebel High Interactivity (HI) API test objects, are not parameterized. For details on these types of objects, see the **Siebel** section of the *HP Unified Functional Testing Add-ins Guide*.

- The name of the parameter that UFT creates for each method argument is in the format **TestObjectName_ArgumentName**.
  - If a parameter with this name already exists for the relevant parameter type, UFT appends an underscore and a number to the end of the new parameter to create a unique name.
  - If the test object name contains characters that are not valid for parameter names or if it contains multi-byte characters, then the test object class is used instead of the test object name.
- The automatic parameterization option is relevant only for tests. UFT does not automatically parameterize steps after you record steps in a component.
- If you select the **Automatically generate With statements after recording** option in addition to the **Automatically parameterize steps** option, then when you stop a recording session, UFT converts the steps to **With** statements and then parameterizes the operation arguments within them.

Guidelines for Automatic Parameterization of Data Table Parameters

- When you use **Global Data Table Parameters** for the automatic parameterization option, the arguments are parameterized using Global Data Table parameters. The constant value that the parameter replaces is stored in the created column in the Global data sheet.

  If the Global data sheet already contains more than one row, the value is entered in all rows of the created Data Table column.
- Data pane sheets can contain 256 columns. At the end of a recording session, UFT parameterizes the relevant operation arguments until it reaches the 256th column. If UFT reaches the maximum number of data table parameters, it stops parameterizing at that point and a message informs you that not all relevant steps were parameterized.
- If the test uses an external data table, and the data table file is read-only, locked by another user, or checked in to ALM version control, the automatic parameterization is not performed and a message is displayed when you finish your recording session.

Guidelines for Automatic Parameterization of Test Parameters

- When you use **Test Parameters** for the automatic parameterization option:
  - Each relevant method argument is converted to a new action parameter.
  - A corresponding test parameter is also created.
  - The constant value that the parameter replaces is stored as the default value for the action

parameter (in the Action Properties dialog box) as well as for the test parameter (in the Parameters pane of the Test Settings dialog box).

For **top-level actions**, UFT also promotes (parameterizes) the action parameter value (in the "Action Call Properties Dialog Box" (described on page 895)) to use the corresponding test parameter for you.

For **nested actions**, UFT creates an action parameter and a corresponding test parameter. However, the action parameter value (in the "Action Call Properties Dialog Box") is not promoted (parameterized) to use a parent action parameter. It is set to use the constant value that it replaced.

When working with nested actions, if you want to use the created test parameter in order to provide the actual values from an external application that calls your test, such as from SAP eCATT, you must manually create corresponding action parameters for all actions that call your action, and then pass (promote) the action parameter values up to each parent action and then to the test parameter level. For details on passing values between action and test parameters, see "How to Use Action Parameters - Use-Case Scenario" on page 893.

Data Driver

Relevant for: GUI tests only

The Data Driver enables you to quickly parameterize several (or all) property values for test objects, checkpoints, and/or method arguments containing the same constant value within a given action.

You can choose to replace all occurrences of a selected constant value with a parameter, in the same way that you can use a **Find and Replace All** operation instead of a step-by-step **Find and Replace** process.

UFT can also show you each occurrence of the constant so that you can decide whether or not to parameterize the value.

Note:

- When finding multiple occurrences of a selected value, UFT conducts a search that is case sensitive and searches only for exact matches. (It does not find values that include the selected value as part of a longer string.)
- You cannot use the Data Driver to parameterize the values of arguments for user-defined methods or VBScript functions.

Example of a Parameterized Test

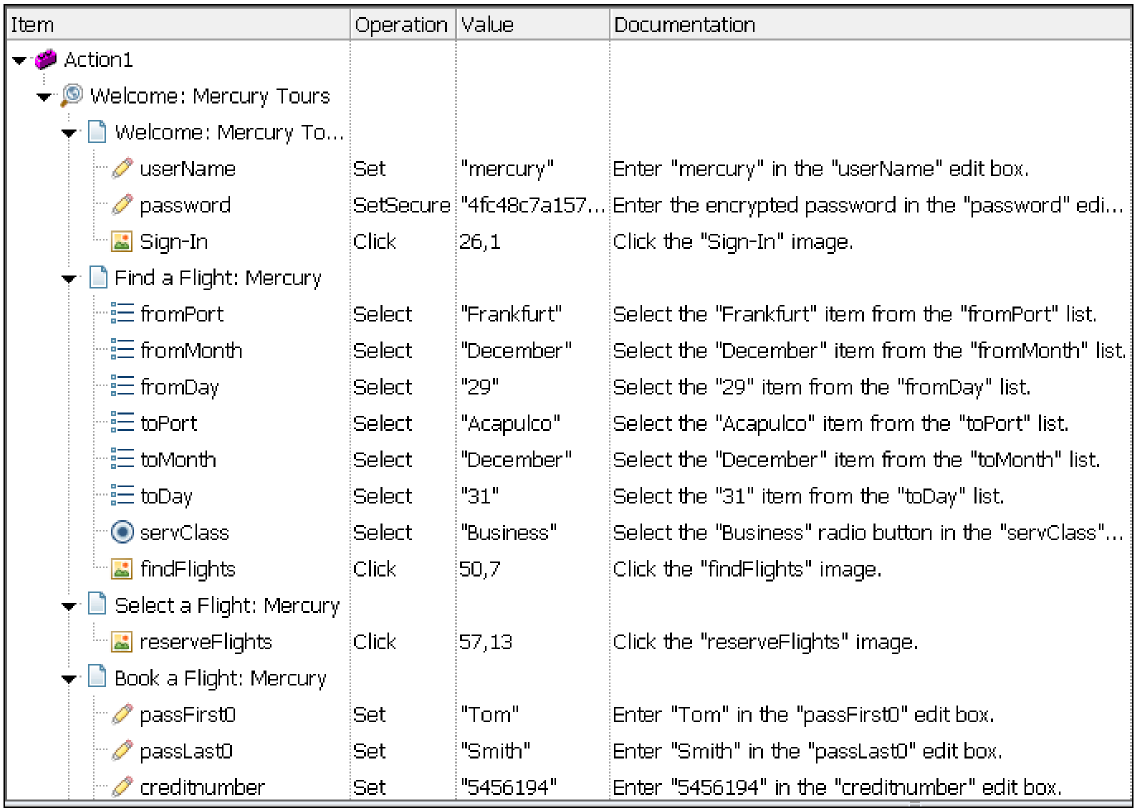
Relevant for: GUI tests only

The following example shows how to parameterize a step method and a checkpoint using data table parameters.

When you test your application, you may want to check how it performs the same operations with multiple sets of data. For example, if you are testing the Mercury Tours sample Web site, you may want to check that the correct departure and the arrival cities are selected before you book a particular flight.

Suppose that you want to check that the flights are booked correctly for a variety of different locations. Rather than create a separate test with a separate checkpoint for each location, you can parameterize the location information. For each iteration of the test, UFT then checks the flight information for the different locations.

The following is a sample test of a flight booking procedure. The departure city is Frankfurt and the arrival city is Acapulco.



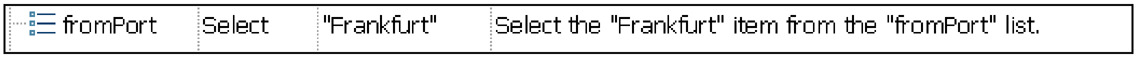
Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury To...			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetSecure	"4fc48c7a157...	Enter the encrypted password in the "password" edi...
Sign-In	Click	26,1	Click the "Sign-In" image.
Find a Flight: Mercury			
fromPort	Select	"Frankfurt"	Select the "Frankfurt" item from the "fromPort" list.
fromMonth	Select	"December"	Select the "December" item from the "fromMonth" list.
fromDay	Select	"29"	Select the "29" item from the "fromDay" list.
toPort	Select	"Acapulco"	Select the "Acapulco" item from the "toPort" list.
toMonth	Select	"December"	Select the "December" item from the "toMonth" list.
toDay	Select	"31"	Select the "31" item from the "toDay" list.
servClass	Select	"Business"	Select the "Business" radio button in the "servClass"...
findFlights	Click	50,7	Click the "findFlights" image.
Select a Flight: Mercury			
reserveFlights	Click	57,13	Click the "reserveFlights" image.
Book a Flight: Mercury			
passFirst0	Set	"Tom"	Enter "Tom" in the "passFirst0" edit box.
passLast0	Set	"Smith"	Enter "Smith" in the "passLast0" edit box.
creditnumber	Set	"5456194"	Enter "5456194" in the "creditnumber" edit box.

This example includes:

Step 1: Parameterize a Step

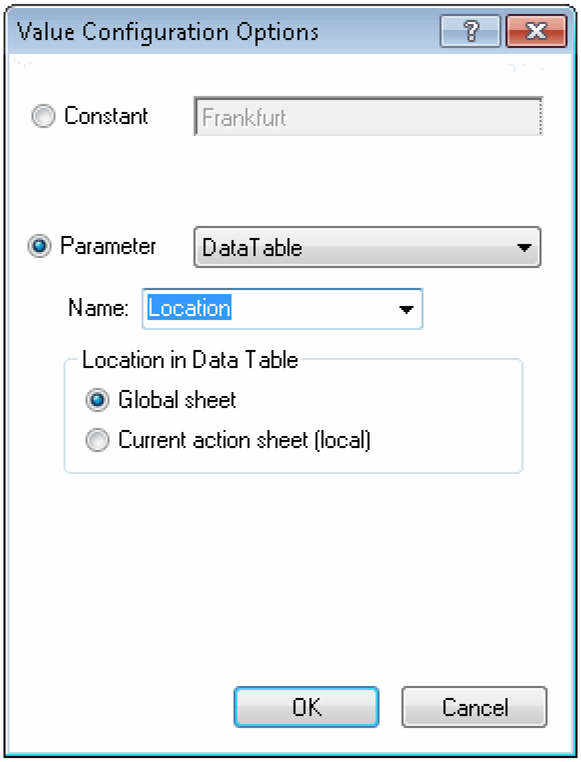
Relevant for: GUI tests only

Parameterize the method argument of the **fromPort** step:



fromPort	Select	"Frankfurt"	Select the "Frankfurt" item from the "fromPort" list.

In the Keyword View, click in the **Value** cell of the step and then click the parameterization icon . In the "Value Configuration Options Dialog Box" (described on page 1579), select the **Parameter** radio button. In the **Name** box, rename p_item to **Location**.



Value Configuration Options

○ Constant: Frankfurt

● Parameter: DataTable

Name: Location

Location in Data Table

● Global sheet
○ Current action sheet (local)

Click **OK**. The **Location** column is added to the Data pane.

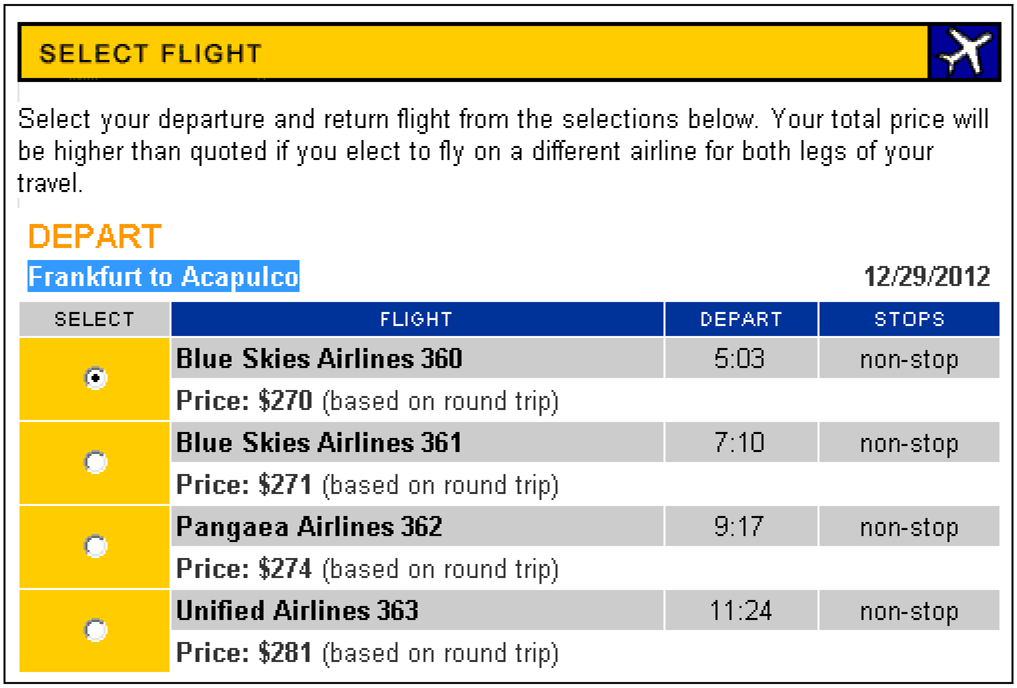
For details on parameterizing a step, see "How to Parameterize Values for Operations or Local Objects" on page 1549.

Step 2: Parameterize a Checkpoint

Relevant for: GUI tests only

Add a parameterized text checkpoint to check that the correct locations were selected before you book a flight. To do this:

Select the **Select a Flight** step. In the Active Screen, highlight the text Frankfurt to Acapulco, right-click and insert a text checkpoint:



SELECT FLIGHT

Select your departure and return flight from the selections below. Your total price will be higher than quoted if you elect to fly on a different airline for both legs of your travel.

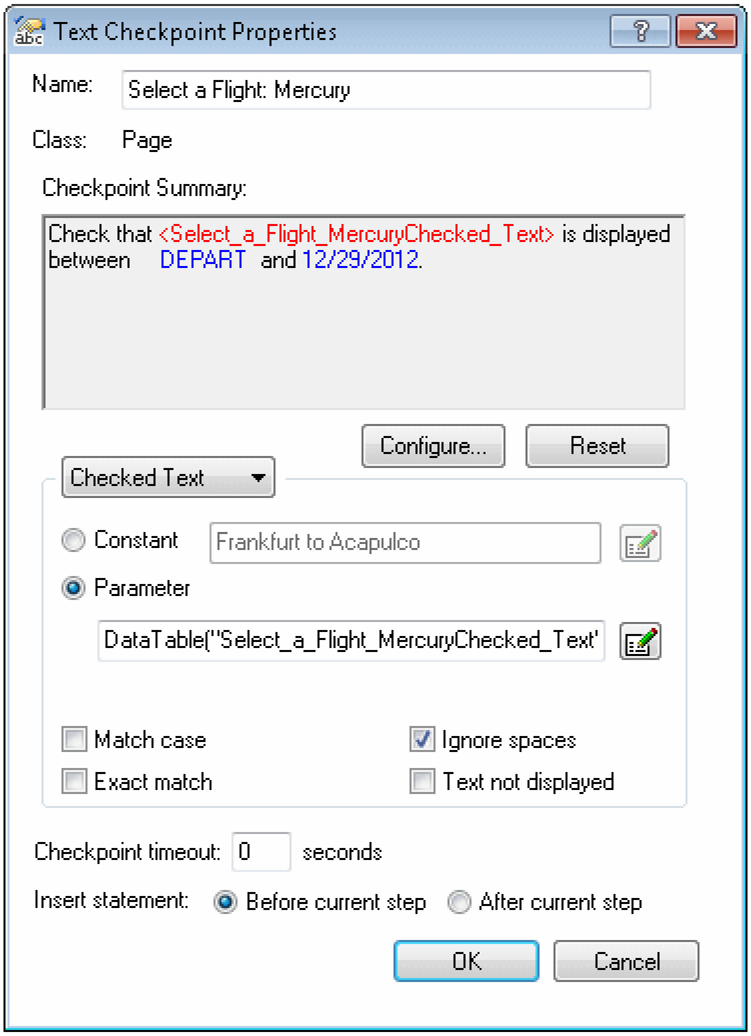
DEPART

Frankfurt to Acapulco 12/29/2012

SELECT	FLIGHT	DEPART	STOPS
●	**Blue Skies Airlines 360**	5:03	non-stop
	Price: $270 (based on round trip)		
○	**Blue Skies Airlines 361**	7:10	non-stop
	Price: $271 (based on round trip)		
○	**Pangaea Airlines 362**	9:17	non-stop
	Price: $274 (based on round trip)		
○	**Unified Airlines 363**	11:24	non-stop
	Price: $281 (based on round trip)		

In the Text / Text Area Checkpoint Properties Dialog Box (described on page 1498), select **Parameter** to parameterize the selected text. Select the **Parameter** radio button and click the **Parameter Options** button .

In the Parameter Options dialog box, rename the data table parameter to Check_Locations_Text. Click **OK** in the Parameter Options dialog box and in the Text Checkpoint Properties dialog box. A **Check_Locations_Text** column is added to the Data pane.



Name: Select a Flight: Mercury

Class: Page

Checkpoint Summary:

Check that <Select_a_Flight_MercuryChecked_Text> is displayed between DEPART and 12/29/2012.

○ Constant Frankfurt to Acapulco

● Parameter

DataTable("Select_a_Flight_MercuryChecked_Text'

□ Match case ☑ Ignore spaces

□ Exact match □ Text not displayed

Checkpoint timeout: 0 seconds

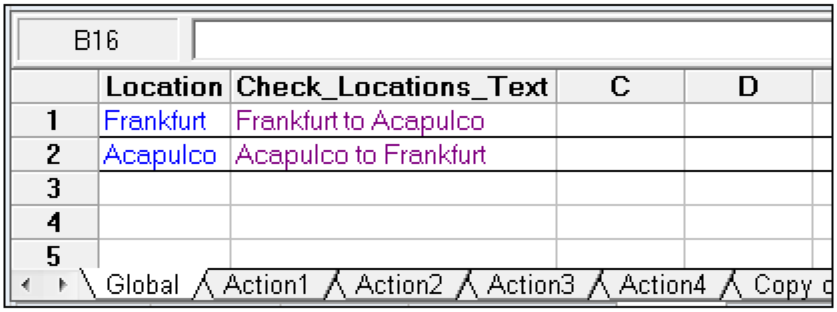
Insert statement: ● Before current step ○ After current step

For details on parameterizing a checkpoint, see "How to Parameterize a Checkpoint Property Value" on page 1550.

Step 3: Enter Data in the Data Pane

Relevant for: GUI tests only

Complete the Data pane, for example:



B16

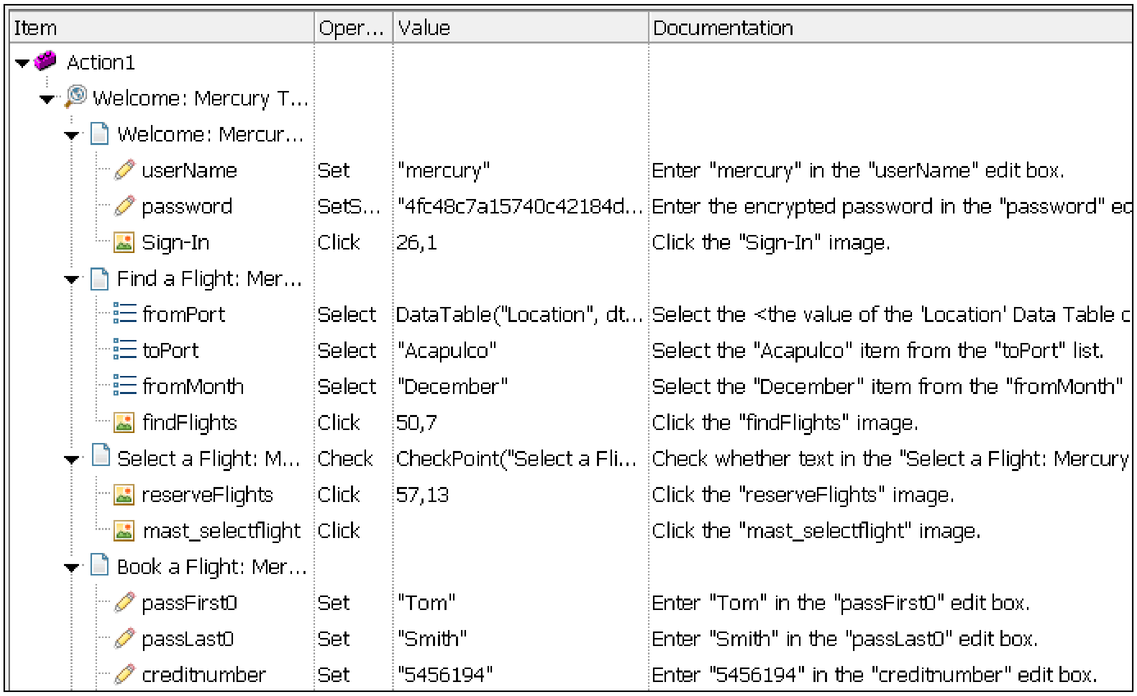
	Location	Check_Locations_Text	C	D
1	Frankfurt	Frankfurt to Acapulco		
2	Acapulco	Acapulco to Frankfurt		
3				
4				
5				

For details on Data pane, see "Data Pane" on page 221.

Step 4: View the Parameterized Test

Relevant for: GUI tests only

The following example shows the test after parameterizing the step and creating a parameterized text checkpoint.



Item	Oper...	Value	Documentation
Action1			
Welcome: Mercury T...			
Welcome: Mercur...			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetS...	"4fc48c7a15740c42184d...	Enter the encrypted password in the "password" ed
Sign-In	Click	26,1	Click the "Sign-In" image.
Find a Flight: Mer...			
fromPort	Select	DataTable("Location", dt...	Select the <the value of the 'Location' Data Table c
toPort	Select	"Acapulco"	Select the "Acapulco" item from the "toPort" list.
fromMonth	Select	"December"	Select the "December" item from the "fromMonth"
findFlights	Click	50,7	Click the "findFlights" image.
Select a Flight: M...	Check	CheckPoint("Select a Fli...	Check whether text in the "Select a Flight: Mercury
reserveFlights	Click	57,13	Click the "reserveFlights" image.
mast_selectflight	Click		Click the "mast_selectflight" image.
Book a Flight: Mer...			
passFirst0	Set	"Tom"	Enter "Tom" in the "passFirst0" edit box.
passLast0	Set	"Smith"	Enter "Smith" in the "passLast0" edit box.
creditnumber	Set	"5456194"	Enter "5456194" in the "creditnumber" edit box.

The parameterized value for the fromPort step is clearly shown as a data table parameter. To see the parameterization setting for the checkpoint, click in the **Value** column for the **Select a Flight** step.

Tasks

How to Parameterize Values for Operations or Local Objects

Relevant for: GUI tests and scripted GUI components

This task describes different ways to parameterize the values for operations in a step or objects in the local object repository.

This task includes:

- "Parameterize a value for an operation using the parameterization icon" below
- "Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test (tests only)" on the next page
- "Enter input and output parameters as values in the Editor (tests only)" on the next page

Parameterize a value for an operation using the parameterization icon

1. To parameterize a value for an operation using the parameterization icon, in the Keyword View click in the **Value** column of the required step.
   - To parameterize property values for local objects, do one of the following:
     - Right-click a step and select **Object Properties**. The "Object Properties Dialog Box" (described on page 1271) opens.
     - Open the "Object Repository Window" (described on page 1274) and select the object.
2. Click the parameterization icon #.
   - To parameterize a value for an operation, in the Keyword View, click the parameterization icon # for the value that you want to parameterize.
   - To parameterize property values for local objects, in the Object Repository click in the **Value** cell for the property that you want to parameterize, and click the parameterization icon #.
   - The Value Configuration Options dialog box opens, showing the currently defined value.
3. Parameterize the value using the "Value Configuration Options Dialog Box" (described on page 1579).

Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test (tests only)

For details, see "Data Driver Dialog Box" on page 1564.

Enter input and output parameters as values in the Editor (tests only)

You can enter input and output parameters as values in the Editor using the Parameter utility object. For details, see "Using Action Parameters in Steps in the Editor" on page 1529.

How to Parameterize a Checkpoint Property Value

Relevant for: GUI tests and scripted GUI components

This task describes how to parameterize the values for properties in checkpoints. You can parameterize the values for properties in checkpoints in one of the following ways:

- "Parameterize a value for a property in a checkpoint using the Checkpoint Properties dialog box" below
- "Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test (tests only)" below
- "Enter input and output parameters as values in the Editor (tests only)" below

Parameterize a value for a property in a checkpoint using the Checkpoint Properties dialog box

1. Open the dialog box for the checkpoint properties.

   Open the dialog box for the checkpoint properties in one of the following ways:

   - Right-click the checkpoint and select **Checkpoint Properties**.
   - Open the "Object Repository Window" (described on page 1274) and select the checkpoint.

2. Use the options in the Configure Value area to parameterize the value.

   a. Select the property whose value you want to parameterize from the table of properties.

   b. In the **Configure value** area, select **Parameter**.

   c. Click the **Parameter Options** button . The "Parameter Options Dialog Box" on page 1556 opens:

Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test (tests only)

For details, see "Data Driver Dialog Box" on page 1564.

Enter input and output parameters as values in the Editor (tests only)

You can enter input and output parameters as values in the Editor using the Parameter utility object.

For details, see "Using Action Parameters in Steps in the Editor" on page 1529.

How to Use User-Defined External Environment Variables

Relevant for: GUI tests only

This task describes how to create an external variable file and use it in your test.

This task includes the following steps:

- "Create an external environment variables file" below
- "(ALM users only) Upload the environment resource file to your ALM project" below
- "Use environment variables in your test" below
- "Results" on the next page

1. **Create an external environment variables file**

   You can create an external environment variable file using the Test Settings dialog box or manually. For details, see "Create an external environment variable file manually" on the next page.

2. **(ALM users only) Upload the environment resource file to your ALM project**

   a. In the ALM Test Resources module, create a new environment variable resource and then upload the .xml environment variable file you want to use with your test. For details, see the *HP Application Lifecycle Management User Guide*.

   b. In UFT, connect to the ALM project. For details, see "ALM Connection Dialog Box" on page 737.

3. **Use environment variables in your test**

   a. Open the Environment pane in the Test Settings dialog box (**File > Settings > Environment** node). For details, see "Environment Pane (Test Settings Dialog Box)" on page 608.

   b. Select **User-defined** from the **Variable type** list.

   c. Select the **Load variables and values from external file (reloaded each run session)** check box.

   d. Use the browse button to the right of the **File** edit box, or enter the full path of the external environment variables file you want to use with your test. If your test is stored in ALM, you must select an file that is stored with your ALM project.

      The variables defined in the selected file are displayed in the list of user-defined environment variables.

4. **Results**

   You can now select the variables in the active file as external user-defined environment parameters in your test. For details, see "Parameter Options Dialog Box" on page 1556.

How to Create an External Environment Variables File

Relevant for: GUI tests only

> **Note:** This task is part of a higher-level task. For details, see "How to Use User-Defined External Environment Variables" on the previous page.

You create an external environment variables file in one of the following ways:

- "Create an external environment variable file using the Test Settings dialog box" below
- "Create an external environment variable file manually" below

Create an external environment variable file using the Test Settings dialog box

1. Define the variables in the Environment pane of the Test Settings dialog box (**File > Settings > Environment** node).

2. Click **Export** to export the user-defined environment variables to an .xml file.

   For details, see "Environment Pane (Test Settings Dialog Box)" on page 608.

Create an external environment variable file manually

1. Create an .xml file using the editor of your choice.

   You can use the UFT environment variable file schema in:<Unified Functional Testing installation folder>\help\
   QTEnvironment.xsd or follow the formatting instructions below.

2. Enter <Environment> on the first line.

3. Enter each variable name-value pair in between <Variable> elements using the following format:

```
<Variable>
   <Name>This is the first variable's name</Name>
   <Value>This is the first variable's value</Value>
   <Description> This text is optional and can be used to add comments.
   It is shown only in the XML and not in UFT</Description>
</Variable>
```

4. Enter </Environment> on the last line.

   Example

   ```
   <Environment>
      <Variable>
         <Name>Address1</Name>
         <Value>25 Yellow Road</Value>
      </Variable>
      <Variable>
         <Name>Address2</Name>
         <Value>Greenville</Value>
      </Variable>
      <Variable>
         <Name>Name</Name>
         <Value>John Brown</Value>
      </Variable>
      <Variable>
         <Name>Telephone</Name>
         <Value>1-123-12345678</Value>
      </Variable>
   </Environment>
   ```

5. Save the file in a location that is accessible from the UFT computer. The file must be in .xml format with an .xml file extension.

Reference

Default Parameter Values

Relevant for: GUI tests only

When you select a value that has not yet been parameterized, UFT generates a default parameter definition for the value. The following table describes how the default parameter settings are determined:

When parameterizing	Condition	Default parameter type	Default parameter name
A value for a step or a checkpoint in an action	At least one input action parameter is defined in the current action	Action parameter	The first input parameter displayed in the "Parameter Values Tab (Action Call Properties Dialog Box)" (described on page 897)
An input action parameter value for a nested action	At least one input action parameter is defined for the action calling the nested action	Action parameter	The first input parameter displayed in the "Parameter Values Tab (Action Call Properties Dialog Box)" (described on page 897) of the calling action
An input action parameter value for a top-level action call	At least one input parameter is defined for the test	Test parameter	The first input parameter displayed in the "Parameter Values Tab (Action Call Properties Dialog Box)" (described on page 897)

If the relevant condition described above is not true, the default parameter type is Data Pane. If you accept the default parameter details, UFT creates a new data table parameter with a name based on the selected value. data table parameters are created in the Global sheet.

For details on Data pane sheets, see "Data Pane" on page 221.

Built-in Environment Variables

Relevant for: GUI tests and scripted GUI components

The following built-in environment variables are available:

Name	Description
ActionIteration (GUI tests only)	The action iteration currently running.
ActionName	The action name.

Name	Description
ControllerHostName	The name of the controller's computer. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
GroupName	The name of the group in the running scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
LocalHostName	The local host name.
OS	The operating system.
OSVersion	The operating system version.
ProductDir	The folder path where the product is installed.
ProductName	The product name.
ProductVer	The product version.
ResultDir	The path of the folder in which the current run results are located. **Note:** You cannot use the **ResultDir** environment variable when running a test from Business Availability Center, LoadRunner, or the Silent Test Runner in UFT.
ScenarioId	The identification number of the scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
SystemTempDir	The system temporary folder.
TestDir	The path of the folder in which the test or scripted component is located.
TestIteration **(GUI tests only)**	The test or iteration currently running.
TestName	The name of the test or scripted component.
UpdatingActiveScreen	Indicates whether the Active Screen images and values are being updated during the update run process. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.
UpdatingCheckpoints	Indicates whether checkpoints are being updated during the update run process. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.
UpdatingTODescriptions	Indicates whether the set of properties used to identify test objects are being updated during the update run process. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

Name	Description
UserName	The Windows login user name.
VuserId	The Vuser identification under load. This variable is relevant only when running as a GUI VUser from the LoadRunner controller.

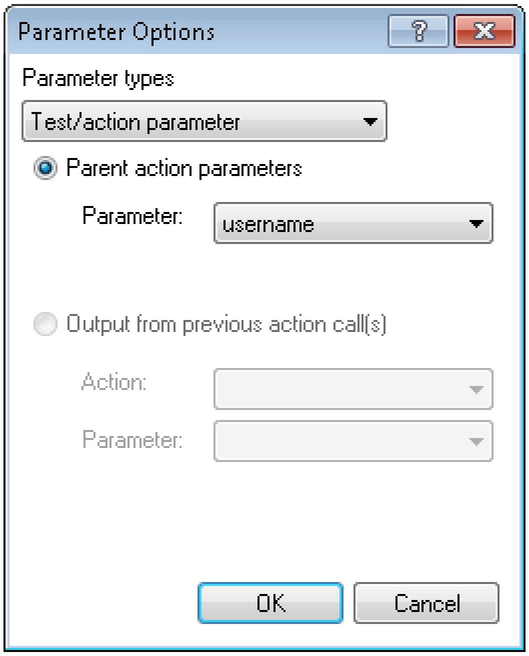
For more details, see the section on built-in environment variables in "Environment Variable Parameters" on page 1536.

Parameter Options Dialog Box

Relevant for: GUI tests and scripted GUI components

This dialog box enables you to define settings for a test/action parameter, a data table parameter, an environment parameter, or a random number parameter.

The example below shows the Parameter Options dialog box for a test/action parameter.



Parameter types: Test/action parameter

● Parent action parameters

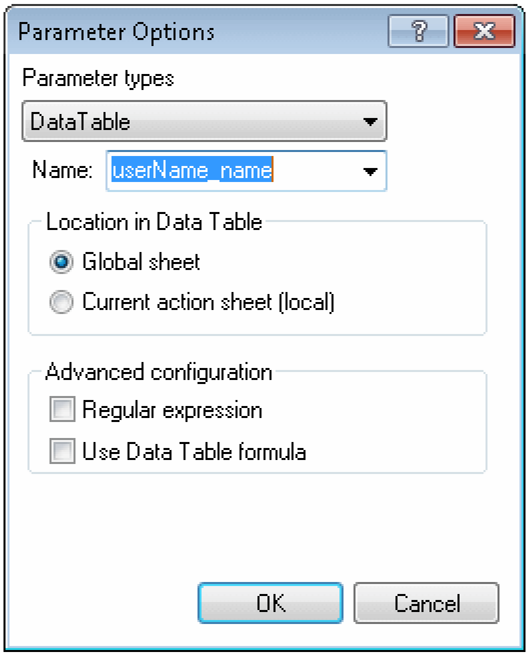
Parameter: username

○ Output from previous action call(s)

Action:

Parameter:

The example below shows the Parameter Options dialog box for a data table parameter.



Parameter types

DataTable

Name: userName_name

Location in Data Table

● Global sheet

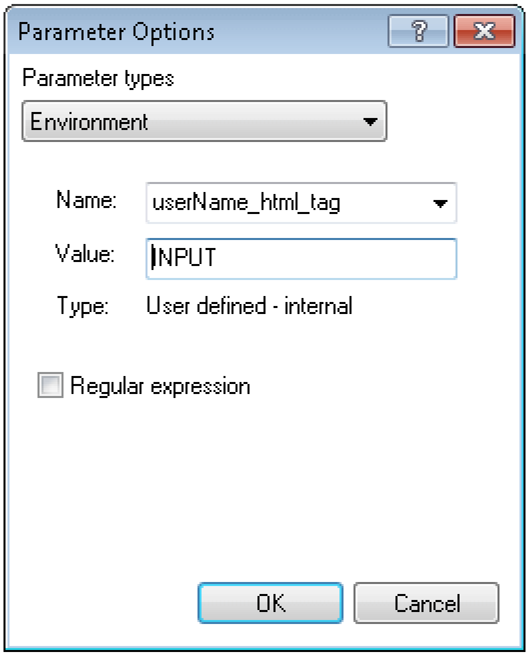
○ Current action sheet (local)

Advanced configuration

□ Regular expression

□ Use Data Table formula

The example below shows the Parameter Options dialog box for a test/action parameter.



Parameter types

Environment

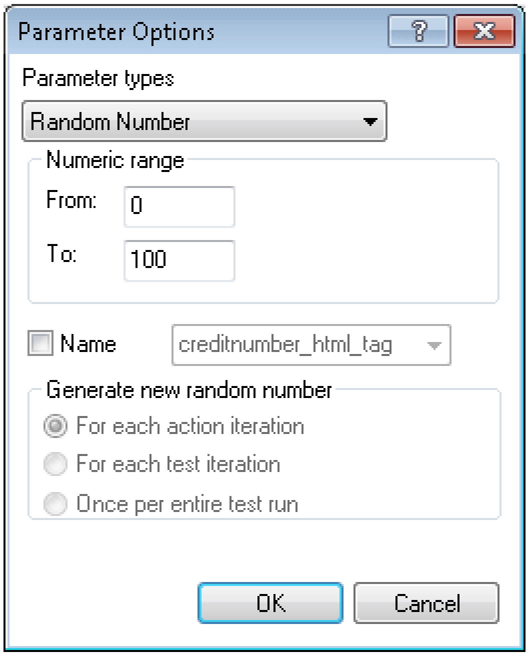
Name: userName_html_tag

Value: INPUT

Type: User defined - internal

□ Regular expression

The example below shows the Parameter Options dialog box for a random number parameter.



Parameter types: Random Number

Numeric range
From: 0
To: 100

□ Name: creditnumber_html_tag

Generate new random number
● For each action iteration
○ For each test iteration
○ Once per entire test run

To access	1. Ensure that a GUI action is in focus in the document pane. 2. Use one of the following: ▪ **For argument values:** In the Keyword View, click the **Configure the value** button . ▪ **For object property values:** In the "Object Repository Window" (described on page 1274), select the object property value you want to parameterize and click the **Configure the value** button . ▪ **For checkpoints:** In the **Configure value** area of the **Checkpoint Properties** dialog box, select the **Parameter** radio button and click the **Parameter Options** button . ▪ **For output value storage locations:** In the "Object Repository Window" (described on page 1274), select the output object property value you want to parameterize and click the **Modify** button in the **Configure value** area. **Note:** This dialog appears when using a scripted component only for checkpoints.

Important information	• You can also use test or action parameter variables using parameterization objects and methods in the Editor. For details, see the *HP UFT Object Model Reference for GUI Testing*. • You cannot select **Use Data Table formula** if **Regular expression** is selected. • The value of an environment variable remains the same throughout the test run, regardless of the number of iterations, unless you change the value of the variable programmatically in your script. • Random number parameters are not appropriate for non-numeric values, such as text or hypertext links. • If you select an existing parameter in your test, when you modify the settings in this dialog box, all instances of that parameter are affected.
Relevant tasks	• "How to Parameterize Values for Operations or Local Objects" on page 1549 • "How to Parameterize a Checkpoint Property Value" on page 1550 • "How to Use User-Defined External Environment Variables" on page 1551
See also	• "Parameterizing Object Values" on page 1526 • "Test and Action Input Parameters" on page 1528 • "Formulas in Data Tables for GUI Testing" on page 228 • "Environment Variable Parameters" on page 1536

User interface elements are described below:

When working with a Test/action parameter

UI Element	Description
Parameter types	The type of parameter you want to use for the value. You can choose from **Test/action parameter**, **Data Table parameter**, **Environment**, or **Random Number**.
Parent action parameter/Test parameter (test and action parameters only)	Select this radio button if you want the parameter to take its value from an input parameter. The available radio button option depends on where you are defining the parameter: • **Parent action parameter.** Available for nested actions and all steps. • **Test parameter.** Available for top-level actions only.

UI Element	Description
Parameter	Specifies the name of the input parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined input parameters for the action.
Output from previous action call(s) (test and action parameters only)	Select this radio button if you want the parameter to take its value from an output parameter. You can select from the output parameters of any previous action in the same hierarchical level as the current action, for which output parameters are defined. • **Action.** Specifies the previous action from which you can choose an output parameter. You can choose any action in the list. • **Parameter.** Specifies the name of the output parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters from the previous action(s).

When working with a Data Table parameter

UI Elements	Description
Parameter types	The type of parameter you want to use for the value. Make sure that **Data Table** is selected.
Name	The name of the parameter to use. The following options are available: • To use an existing parameter, select it from the list. • To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter.
Location in Data Table	• **Global sheet.** You store data in the Global tab when you want it to be available to all actions in your test and you want the data to control the number of test iterations. • **Current action sheet (local).** You store data in the action's tab when you want to use the data in data table parameters for that action only and you want the data to control the number of action iterations. For details, see "When to Choose Global or Action Data Table Parameters " on page 1538.
Regular expression	Enables you to set the value of the parameter as a regular expression. **Note:** This option is available only when parameterizing checkpoint and object property values.

UI Elements	Description
Use Data Table formula	Enables you to use formulas predefined in the Data Table. For details on setting formulas, see "Formulas in Data Tables for GUI Testing" on page 228. For checkpoints, UFT inserts two columns in the Data pane. The first column contains a formula that checks the validity of the output in the second column. UFT uses the data in the output column to compute the formula, and inserts a value of TRUE or FALSE in the table cell of the formula column.

When working with an Environment parameter

UI Elements	Description
Parameter types	The type of parameter you want to use for the value. Make sure that **Environment** is selected.
Name	The name of the parameter. For an internal user-defined environment variable parameter, you can create a new parameter by using the default parameter name or entering a new, descriptive name. Alternatively, you can select an existing internal user-defined environment variable parameter from the list. **Note:** • If you edit the name displayed in the **Name** box for an existing parameter, you create a new internal user-defined environment variable parameter. The original environment variable parameter is not modified. For details on modifying existing environment variable parameters, see "Environment Pane (Test Settings Dialog Box)" on page 608. • If you are parameterizing an argument that receives a predefined constant or number, only the environment variable parameters whose value is of type integer are shown in the **Name** list.
Value	Specifies the value of the parameter. You can enter the value for a new user-defined internal parameter, or modify the value for an existing user-defined internal parameter. External and built-in environment variable parameter values cannot be modified in this dialog box. If the entire value of a selected environment variable parameter cannot be displayed in the **Value** box, it is shown as **[complex value]**. For example, the value of a list's **all items** property is a multi-line value, where each line contains the value of an item in the list. You can view or edit a complex value by clicking the View/Edit Complex Value button 📝. For details, see "Edit Complex Value Dialog Box " on page 1563.

UI Elements	Description
Type	The type of environment variable parameter (read-only): - **internal user-defined** - **external user-defined** - **built-in**
Regular expression	The value of the parameter as a regular expression. This option is available only when parameterizing a checkpoint or object property text string value, and the selected environment variable parameter type is **internal user-defined**. For details on regular expressions, see "Regular Expressions Overview" on page 318.

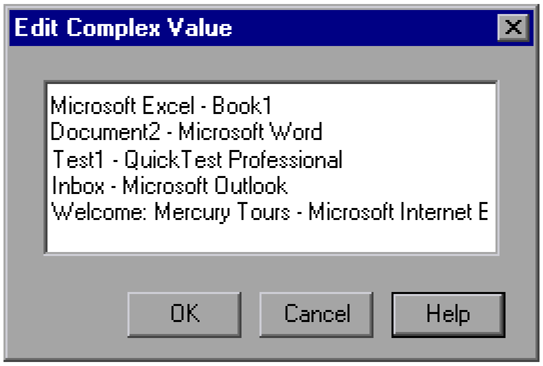
When working with a Random Number parameter

UI Elements	Description
Parameter types	The type of parameter you want to use for the value. Make sure that **Random Number** is selected.
Numeric range	The range from which the random number is generated. You can modify the range by entering different values in the **From** and **To** boxes. **Default range:** 0 to 100 **Minimum From value:** 0 **Maximum To value:** 2147483647
Name	The name of your parameter. Assigning a name to a random parameter enables you to use the same parameter several times in your test. You can select an existing named parameter or create a new named parameter by entering a new, descriptive name.
Generate new random number	The generation timing for a named random parameter. This box is enabled when you select the **Name** check box. You can select one of the following options: - **For each action iteration.** Generates a new number for each action iteration. - **For each test iteration.** Generates a new number for each global iteration. - **Once per entire test run.** Generates a new number the first time the parameter is used. The same number is used for the parameter throughout the run session.

Edit Complex Value Dialog Box

Relevant for: GUI tests only

This dialog box enables you to edit a parameter value that cannot be displayed in the Parameter Options dialog box.



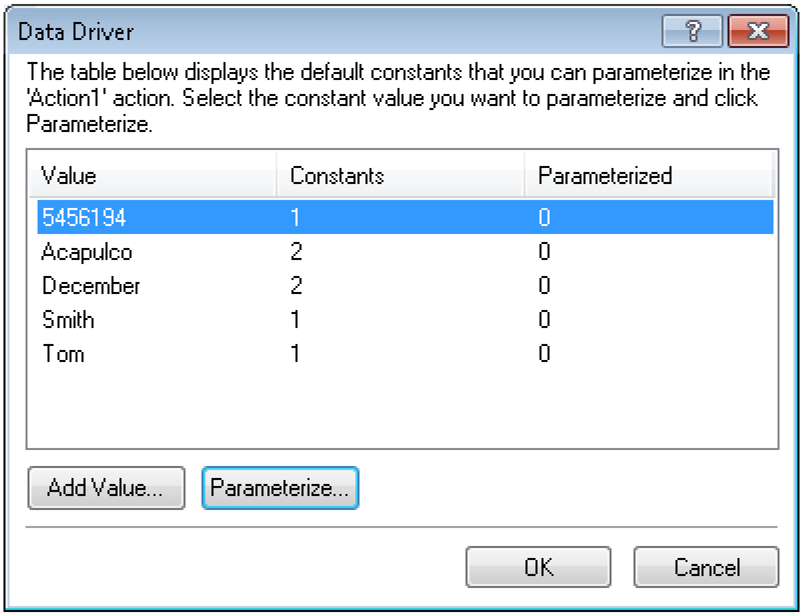
Microsoft Excel - Book1
Document2 - Microsoft Word
Test1 - QuickTest Professional
Inbox - Microsoft Outlook
Welcome: Mercury Tours - Microsoft Internet E

To access	The **Parameters Options** dialog box displays the value of the parameter in the **Value** box. If the value cannot be entirely displayed in the **Value** box (such as a list), click the **View/Edit Complex Value** button to open the **Edit Complex Value** dialog box.
Important information	• **Internal user-defined environment variable parameter.** You can edit the value. • **External or built-in environment variable parameter.** You can view the value but you cannot modify it in this dialog box.
Relevant tasks	"How to Use User-Defined External Environment Variables" on page 1551
See also	"Environment Variable Parameters" on page 1536

Data Driver Dialog Box

Relevant for: GUI tests only

This dialog box displays a list of all the default constants for an action. For each constant value you add to the list, it displays the number of times the constant valuEWe appears in the action.



The table below displays the default constants that you can parameterize in the 'Action1' action. Select the constant value you want to parameterize and click Parameterize.

Value	Constants	Parameterized
5456194	1	0
Acapulco	2	0
December	2	0
Smith	1	0
Tom	1	0

To access	1. Do one of the following: ▪ Ensure that a GUI action is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test node or one of its child nodes. 2. Select **Tools > Data Driver**.
Relevant tasks	• "How to Parameterize Values for Operations or Local Objects" on page 1549 • "How to Parameterize a Checkpoint Property Value" on page 1550
See also	"Data Driver" on page 1543

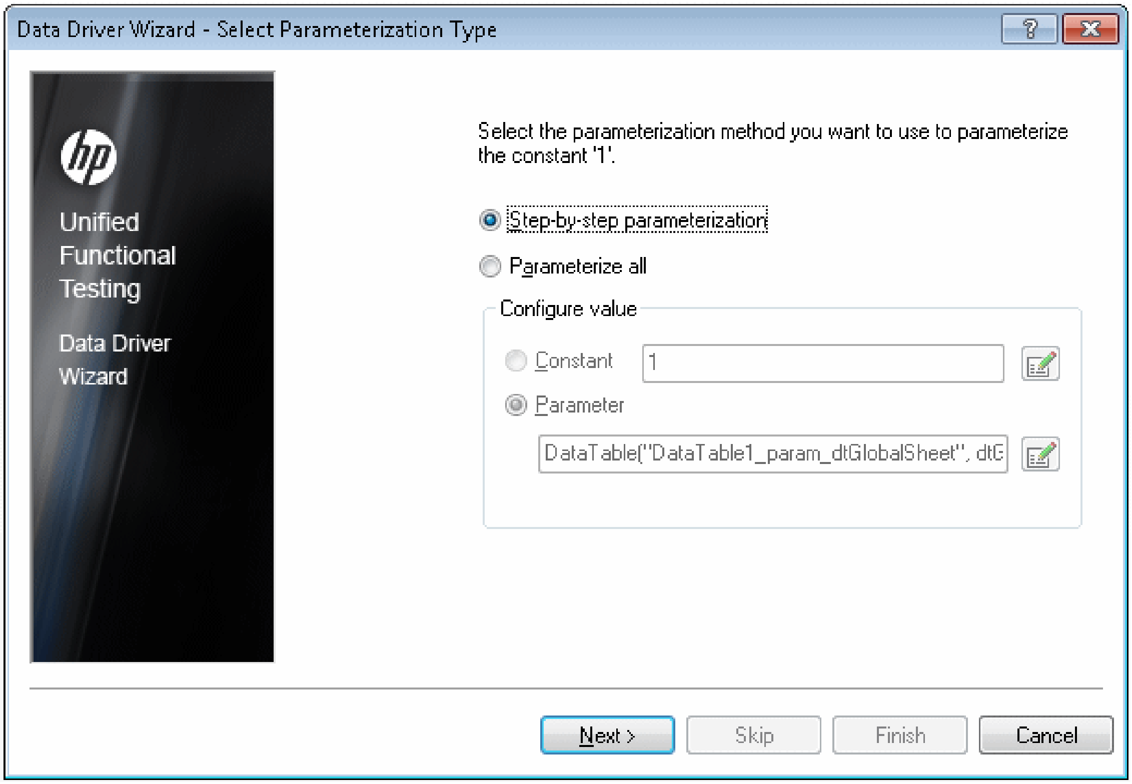
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<constants list>	The list of default constant values as well as any constants you added during the current Data Driver session. The list also displays the number of times the constant value occurs in the action, and the number of occurrences that are set to be parametrized.
Add Value	Opens the Add Constant dialog box, which enables you to add a valid constant to the Constants list. Any object property or checkpoint value that exists in your action is a valid constant value. If you chose not to wait for the constants to load you can use this option to add the constant values that you want to parameterize to the Data Driver.
Parameterize	Opens the Data Driver Wizard, which enables you to parameterize all occurrences of the selected value. You can set the parameterization for each occurrence step-by-step, or you can parameterize all occurrences in a single operation.

Data Driver Wizard - Select Parameterization Type Page

Relevant for: GUI tests only

This wizard enables you to select the parameterization method, and set parameterization options if you select the **Parameterize all** method.



Data Driver Wizard - Select Parameterization Type

Select the parameterization method you want to use to parameterize the constant '1'.

● Step-by-step parameterization

○ Parameterize all

Configure value

○ Constant: 1

● Parameter: DataTable("DataTable1_param_dtGlobalSheet", dtG

To access	1. Do one of the following: ▪ Ensure that a GUI test or action is in focus in the document pane. ▪ In the Solution Explorer, select a GUI test node or one of its child nodes. 2. Select **Tools > Data Driver** and select the **Parameterize** button in the Data Driver dialog box.
Relevant tasks	• "How to Parameterize Values for Operations or Local Objects" on page 1549 • "How to Parameterize a Checkpoint Property Value" on page 1550
Wizard map	This wizard contains: **Data Driver Wizard - Select Parameterization Type Page** > "Data Driver Wizard - Parameterize the Selected Step Page"(page 1567)

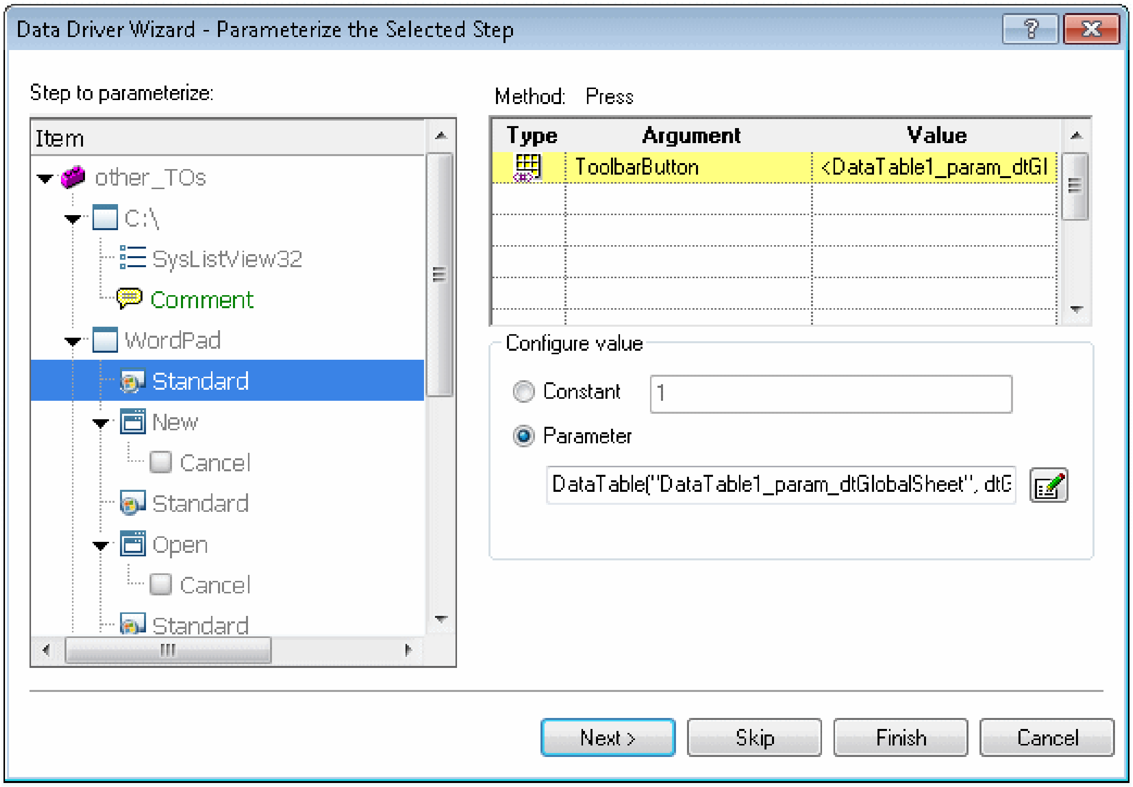
User interface elements are described below:

UI Elements	Description
Step-by-step parameterization	Enables you to view the current values of each step containing the selected value. For each step, you can choose whether or not to parameterize the value, and if so, which parameterization options you want to use. The **Next** button is enabled when you select this option.
Parameterize all	Enables you to parameterize all occurrences of the selected value throughout the action. When you select this option the **Configure value** area is enabled. The **Finish** button is enabled when you select this option.
Configure value	Enables you to set the parameterization options you want to apply to all occurrences of the selected value. Select your parameterization preferences the same way that you would for an individual step. For details on setting parameterization options, see "Configure Value Area " on page 1575.
Next	Opens the "Data Driver Wizard - Parameterize the Selected Step Page". (Enabled only when **Step-by-step parameterization** is selected.)
Skip	Disabled for this page.
Finish	Saves your parameterization options and closes the Data Driver Wizard. The "Data Driver Dialog Box" displays the number of occurrences of the selected value to parameterize. **Note:** Your parameterization settings are not applied until you click **OK** on the Data Driver main page.

Data Driver Wizard - Parameterize the Selected Step Page

Relevant for: GUI tests only

This wizard page displays the steps in the current action and enables you to set your parameterization preferences for the selected step.



Wizard map	The Data Driver Wizard contains: "Data Driver Wizard - Select Parameterization Type Page" (page 1565) > **Data Driver Wizard - Parameterize the Selected Step Page**
See also	"Data Driver" on page 1543

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

UI Elements	Description
Step to parameterize	Displays the steps in the current action. Highlights the first step with an object property or checkpoint value containing the constant value selected in the "Data Driver Dialog Box".
<Step information>	The top right part of the dialog box displays information about the selected object property or checkpoint in the **"Step to parameterize"** area.

UI Elements	Description
<Properties/arguments information> (for objects and checkpoints only)	The Properties area displays the property to parameterize within the context of the other properties or arguments that apply to the selected step. **Note:** You can parameterize only the selected value. The other values are displayed for informative purposes only.
Configure value area	The default parameterization settings are selected for the value. To modify the settings click the **Parameter Options** button to set the parameterization options you want to apply to this step. You select your parameterization preferences the same way that you would for an individual step. For details, see "Configure Value Area " on page 1575.
Next	Saves your parameterization preferences for the selected step, and opens the Parameterize the Selected Step page for the next occurrence of the selected value. If there are no remaining occurrences of the value, clicking **Next** opens the Finished page.
Skip	The value of the selected step remains a constant (not parameterized), and the Parameterize the Selected Step page for the next occurrence of the selected value opens. If there are no remaining occurrences of the value, clicking **Skip** opens the Finished page.
Finish	Applies the parameterization settings of the current step to this step and all remaining steps containing the selected value. The Data Driver Wizard closes and the Data Driver main page shows how many occurrences you selected to parameterize and how many remain as constants.

Chapter 53: Value Configuration and Parameterization

Relevant for: GUI tests and components

This chapter includes:

Concepts

Value Configuration Overview

Relevant for: GUI tests and scripted GUI components

UFT enables you to configure the values for properties and other items by defining a value as a constant or a parameter. You can also use regular expressions for some values to increase the flexibility and adaptability of your tests. For details on regular expressions, see "Regular Expressions Overview" on page 318.

Some dialog boxes, such as the Checkpoint Properties dialog boxes, include a **Configure value** area, in which you can define the value for a selected item as a constant or a parameter. In other contexts, such as the Keyword View, Step Generator, and Object Repository window, you can select a value directly and parameterize it or define it as a constant.

- **Constant.** A manually defined value that remains unchanged for the duration of the test. In certain contexts, you can define a constant value using a regular expression.

- **Parameter.** A value that is defined or generated externally and is retrieved during a run session. For example, a parameter value may be defined in an external file or generated by UFT.

When you define a value as a parameter, you can also specify other settings according to the parameter type. For details on using parameters in your tests, see "Parameterizing Object Values" on page 1526.

Working with Local and Component Parameters

Relevant for: Keyword GUI components

You can define input parameters that pass values into your keyword component, and output parameters that pass values from your component to external sources or from one step to another step. You can then use these parameters to parameterize input and output values in steps.

You can define two types of parameters—**local parameters** and **component parameters**.

Local parameter. Variable values defined within a component for use within the same component.

Local input parameter values can be received and used by a later parameterized step in the same component. You define local input parameters in the Keyword View using the "Value Configuration Options Dialog Box" (described on page 1579) and the "Parameters Tab (Properties Pane - GUI Testing)" (described on page 1475).

Local output parameters can be returned by an operation or component step for use within the same component. Local output parameter values can be viewed in the business process run results. You define local output parameters using the Output Options dialog box.

You cannot delete local parameters, but you can cancel the input or output to them.

Component parameter. Variable values defined within a component for use in the same component or later components in the business process test.

Component input parameter values can be received and used as the values for specific, parameterized steps in the component.

Component output parameter values can be returned as input parameters in components that are used later in the test. These values can also be viewed in the business process run results.

You define component input and output parameters in the Parameters pane of the Component Settings dialog box, or in the ALM Business Components module. For details, see "Parameters Tab (Properties Pane - GUI Testing)" on page 394 or the *HP Business Process Testing User Guide*.

> **Tip:** Additionally, if you are working with a business process test or flow, you can define parameters in the Properties pane. For details, see "Parameters Tab (Properties Pane - BPT)" on page 444.

After you define a parameter you can use it to parameterize a value. Alternatively, you can apply a constant value to the parameter by typing it directly in the **Value** cell.

For instructions on how to parameterize input values, see "How to Parameterize Input Values (Keyword Components)" on the next page.

For instructions on how to parameterize output values, see "How to Create or Modify an Output Value Step" on page 1494.

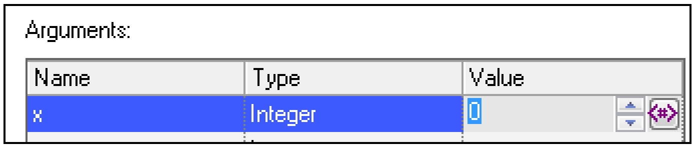
Tasks

How to Configure Constant and Parameter Values

Relevant for: GUI tests and components

This task describes how to define a value as a constant or a parameter:

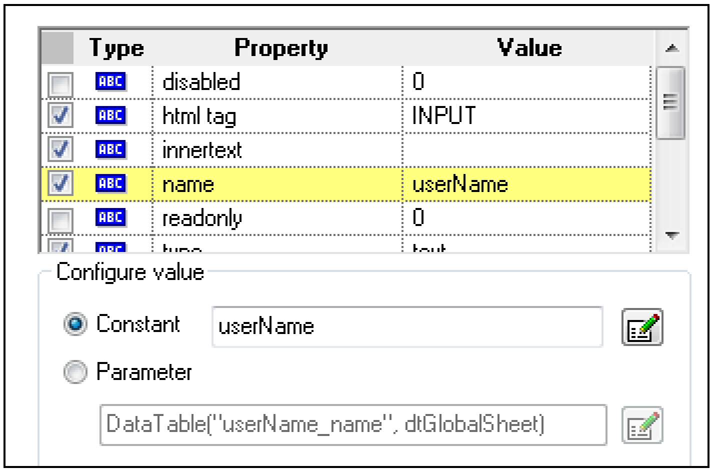
- In the **Value** area (available from the Keyword View, Step Generator, Object Repository window, and so on), click the parameterization button for a selected value to open the "Value Configuration Options Dialog Box" (described on page 1579). For example:



Arguments:

Name	Type	Value
x	Integer	0

- Above the **Configurevalue** area of a dialog box, select a property or argument, for example:



	Type	Property	Value
□	ABC	disabled	0
☑	ABC	html tag	INPUT
☑	ABC	innertext	
☑	ABC	name	userName
□	ABC	readonly	0
☑	ABC	type	text

Configure value

● Constant userName

○ Parameter

DataTable("userName_name", dtGlobalSheet)

Then, in the **Configure value** area, select the **Constant** or **Parameter** radio button and click the **Constant Value Options** or **Parameter Options** button . For details, see "Configure Value Area " on page 1575.

For additional details, see:

- **For tests and scripted components:** "Parameter Options Dialog Box" on page 1556
- **For keyword components:** "Parameter Options Dialog Box (Local and Component Input Parameters for Checkpoints)" on page 1475.

How to Parameterize Input Values (Keyword Components)

Relevant for: Keyword GUI components

This task describes how to parameterize input values for a step using a local parameter or a component parameter.

This task includes the following steps:

- "Open the Value Configuration Options dialog box or Parameter Options dialog box" on the next page
- "Select the type of parameter and define its details" on the next page
- "Results" on page 1575

Open the Value Configuration Options dialog box or Parameter Options dialog box

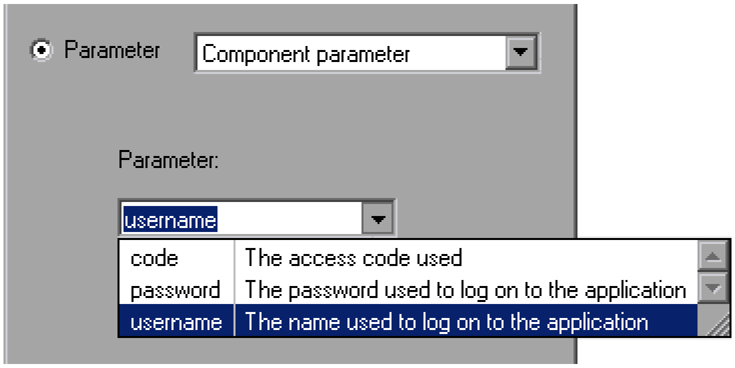
For details, see:

- "Value Configuration Options Dialog Box" on page 1579
- "Parameter Options Dialog Box (Local and Component Input Parameters for Checkpoints)" on page 1475

Select the type of parameter and define its details

1. In the **Parameter** box, select one of the following:

   - **Component parameter.** Select the component parameter you want to use for the parameterized value. The names and full descriptions of the available component parameters are displayed as read-only. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

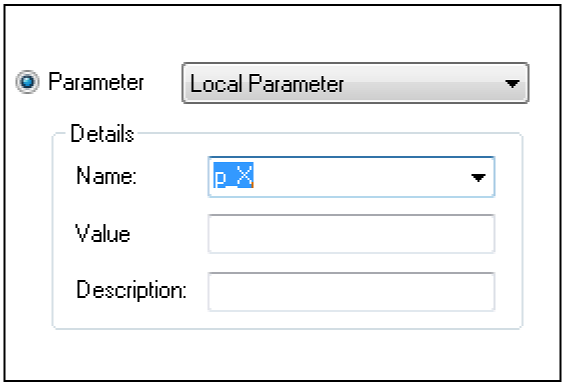


● Parameter: Component parameter

Parameter: username

code	The access code used
password	The password used to log on to the application
username	The name used to log on to the application

   - **Local parameter.** The details for the local parameter type are displayed.



● Parameter: Local Parameter

Details
Name: p_X
Value:
Description:

2. Specify the property details for the local parameter:

   - **Name.** Enter a meaningful name (case-sensitive) for the parameter or choose one from the list.

- **Value.** Enter an input value for the parameter. For details, see "Parameters Tab (Properties Pane - GUI Testing)" on page 394.
- **Description.** Enter a brief description for the parameter.

> **Tip:**
>
> You can cancel the parameterization of a value by selecting the **Constant** option in the "Value Configuration Options Dialog Box" (described on page 1579) and entering a constant value. The Constant box offers the same editing options as the Value cell in which you clicked the parameterization button ⟨#⟩ to open this dialog box.
>
> If you click in the **Value** cell after you define a **component parameter** for it, the 🧩 icon is displayed in each part of the cell for which a component parameter is defined.
>
> If you click in the **Value** cell after you define a **local parameter** for it, the 🏠 icon is displayed in each part of the cell for which a local parameter is defined.

Results

The local or component parameter is displayed in the **Value** cell of your step. When the component runs, it will use the value specified in the parameter for the step.

Reference

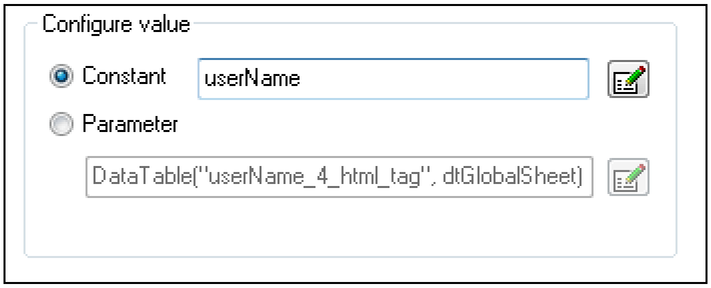
Configure Value Area

Relevant for: GUI tests and components

This area enables you to configure object property values or the values of the operation arguments defined for the step.

The following examples illustrate various value definitions:

Single-line constant



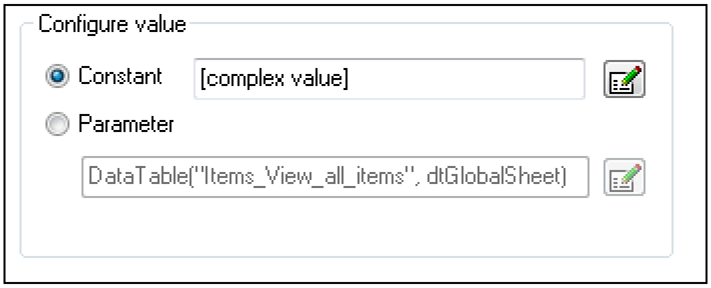
Configure value

● Constant userName

○ Parameter

DataTable("userName_4_html_tag", dtGlobalSheet)

Complex constant (tests only)



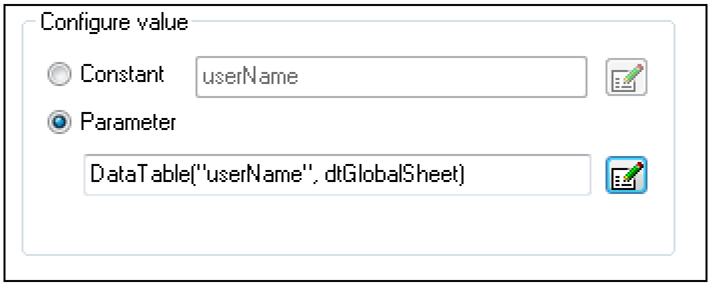
Configure value

● Constant [complex value]

○ Parameter

DataTable("Items_View_all_items", dtGlobalSheet)

Parameter



Configure value

○ Constant userName

● Parameter

DataTable("userName", dtGlobalSheet)

To access	This area is available in many dialog boxes, for example, the various Checkpoint Properties and Output Value dialog boxes, the various Parameter Options dialog boxes, the "Parameterization / Properties Dialog Box (Checkpoints)" (described on page 1473) (for components), the Data Driver Wizard, and so on.

User interface elements are described below:

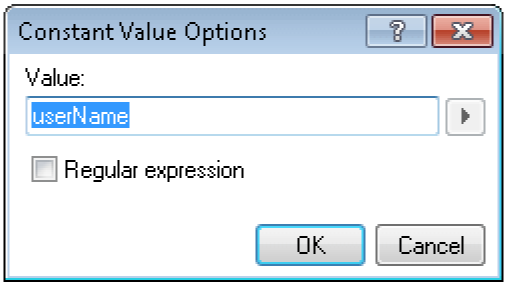
UI Elements	Description
Constant	A manually defined value that remains unchanged for the duration of the run. In certain contexts, you can define a constant value using a regular expression. This is the default option. If the entire value cannot be displayed in the **Constant** box, it is shown as **[complex value]**. For example, the value of a list's **all items** property is a multi-line value, where each line contains the value of an item in the list. You can edit a value directly in the **Constant** box or by clicking the **Constant Value Options** button . This also enables you to define a **string** value or **complex value** as a regular expression.

UI Elements	Description
Parameter	A value that is defined or generated externally and is retrieved during a run session. For example, a parameter value may be defined in an external file or generated by UFT. The **Parameter** box displays: • The current parameter definition for a value that is already parameterized • The default parameter definition for a value that is not yet parameterized. For more details on default parameter definitions, see: ▪ **For tests and scripted components:** "Default Parameter Values" on page 1554 ▪ **For keyword components:** "Value Configuration Options Dialog Box" on page 1579. You can click the **Parameter Options** button to open the Parameter Options dialog box that enables you to select a different parameter type or modify the parameter settings for the value. For details, see "Parameter Options Dialog Box" on page 1556 (if you are working in a test or scripted component) or "Value Configuration Options Dialog Box" on page 1579 (if you are working in a scripted component). For more details on using parameters in GUI tests, see "Parameterizing Object Values" on page 1526.

Constant Value Options Dialog Box

Relevant for: GUI tests and scripted GUI components

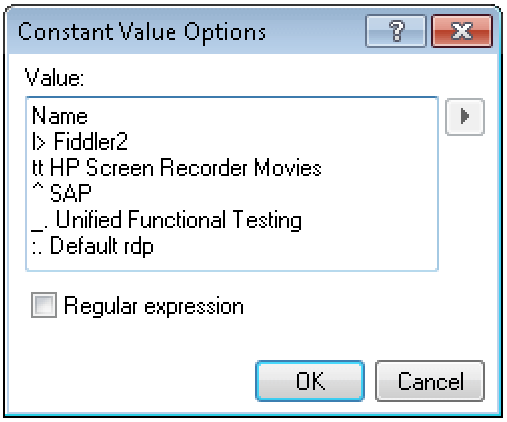
This dialog box enables you to edit the value of a constant.



Value: userName

□ Regular expression

For a **complex value** (a value that cannot be displayed entirely in the **Constant** box in the **Configure value** area of a dialog box), the Constant Value Options dialog box expands to show the entire contents of the value.



```
Name
I> Fiddler2
tt HP Screen Recorder Movies
^ SAP
_. Unified Functional Testing
:. Default rdp
```

To access	Click the **Constant Value Options** button in the **Configure value** area of a dialog box.
See also	"Value Configuration Options Dialog Box" on the next page

User interface elements are described below:

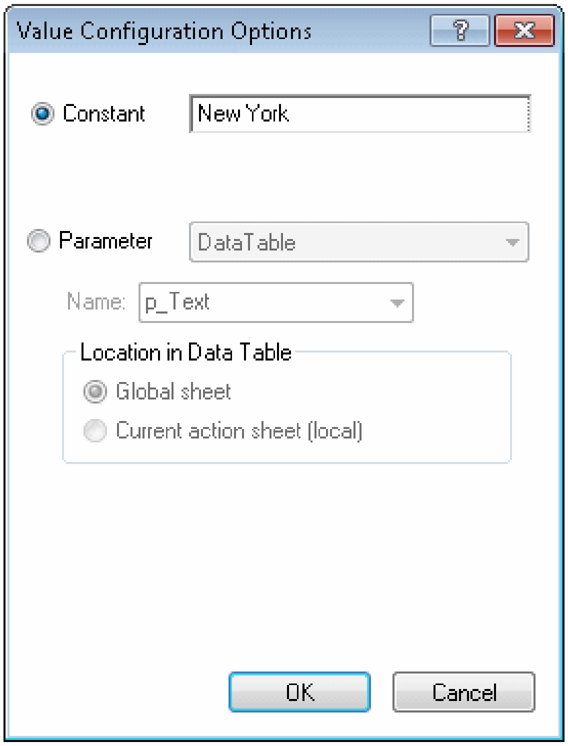
UI Elements	Description
Value	The value for the constant.
Regular expression	Sets the defined value as a regular expression. If you select this option, an arrow is enabled to the right of the **Value** text box, enabling you to insert regular expression characters and to test your regular expression to make sure that it suits your needs. • For general details on regular expressions, see "Regular Expressions Overview" on page 318. • For details on defining a regular expression, see "Regular Expression Characters and Usage Options" on page 352. • For details on inserting regular expression characters directly from a list, see "Smart Regular Expression List" on page 361. • For details on testing your regular expression, see "Regular Expression User Interface" on page 352.

Value Configuration Options Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to define a selected value as a constant or a parameter. In some situations, you can also define the constant or parameter using a regular expression.

The following examples illustrate the Value Configuration Options dialog box with and without the **Regular expression** check box (relevant for tests and scripted components). The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.



Value Configuration Options

● Constant: New York

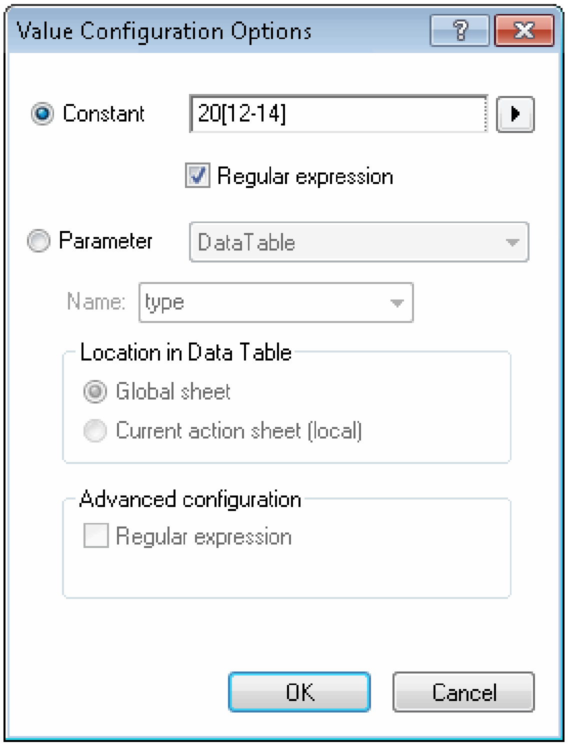
○ Parameter: DataTable

Name: p_Text

Location in Data Table

● Global sheet

○ Current action sheet (local)



Value Configuration Options

● Constant: 20[12-14]

☑ Regular expression

○ Parameter: DataTable

Name: type

Location in Data Table

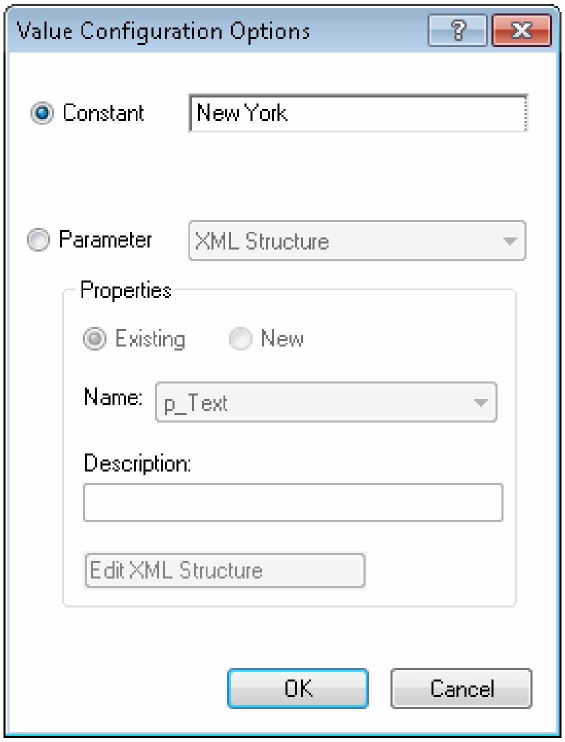
● Global sheet

○ Current action sheet (local)

Advanced configuration

□ Regular expression

The following examples illustrate the Value Configuration Options dialog box (for keyword components) when either **Constant** or **Parameter** value type is selected. The options shown in this dialog box change according to the value type.



Value Configuration Options

● Constant: New York

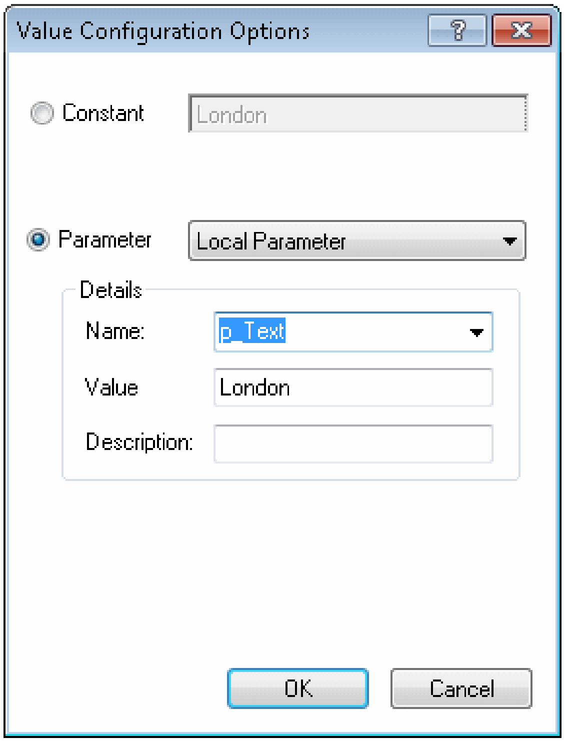
○ Parameter: XML Structure

Properties

● Existing ○ New

Name: p_Text

Description:



Value Configuration Options

○ Constant: London

● Parameter: Local Parameter

Details

Name: p_Text

Value: London

Description:

To access	1. Ensure that a GUI action or component is in focus in the document pane. 2. In the Keyword View, click the parameterization button for a selected value.
Important information	**Relevant for keyword components:** If at least one input component parameter is defined in the component, the default input type is **Component parameter** and the default input name is the first output parameter displayed in the Parameters pane of the Business Component Settings dialog box.

User interface elements are described below:

UI Elements	Description
Constant	A manually defined value that remains unchanged for the duration of the run. You can edit the value directly in the **Constant** box. This box offers the same editing options as the **Value** cell in which you clicked the parameterization button to open this dialog box. For details, see "How to Define Values for Your Step Arguments " on page 928.

UI Elements	Description
Regular expression (tests and scripted components)	Enables you to specify a constant value using a regular expression. This option is available only for string value types in some situations, for example, when parameterizing an object identification property value. This option is not available for method arguments or for keyword components. **To use a regular expression:** Enter a regular expression in the **Constant** box and select the **Regular expression** check box. For more details, see "Regular Expressions Overview" on page 318, "Smart Regular Expression List" on page 361, and "Regular Expression Evaluator" on page 358.
Parameter	A value that is defined or generated externally and is retrieved during a run session. The parameter section displays: • The current parameter type and details for the value if a value is already parameterized. • The default parameter type and details for the value if a value is not yet parameterized. You can change the default definition by selecting a different parameter type or modifying the parameter settings for the value. For details, see: ▪ **For tests and scripted components:** "Default Parameter Values" on page 1554 ▪ **For keyword components:** "Working with Local and Component Parameters" on page 1571. **Options for tests and scripted components include:** The options in the **Parameter** section change according to the parameter type you select and are very similar to the options in the Parameter Options dialog box. For details, see "Parameter Options Dialog Box" on page 1556 For more details on using parameters in your tests or scripted component, see "Parameterizing Object Values" on page 1526.
Parameter (Cont.)	**Options for keyword components include:** • **Component parameter.** Variable values defined within a component for use in the same component or later components in a business process test. • **Local parameter**. Variable values defined within a component for use exclusively within the same component. For details about component parameters and local parameters, see "Working with Local and Component Parameters" on page 1571, "How to Parameterize Input Values (Keyword Components)" on page 1573, and "Parameters Tab (Properties Pane - GUI Testing)" on page 394.

Using Property Patterns to Identify Objects

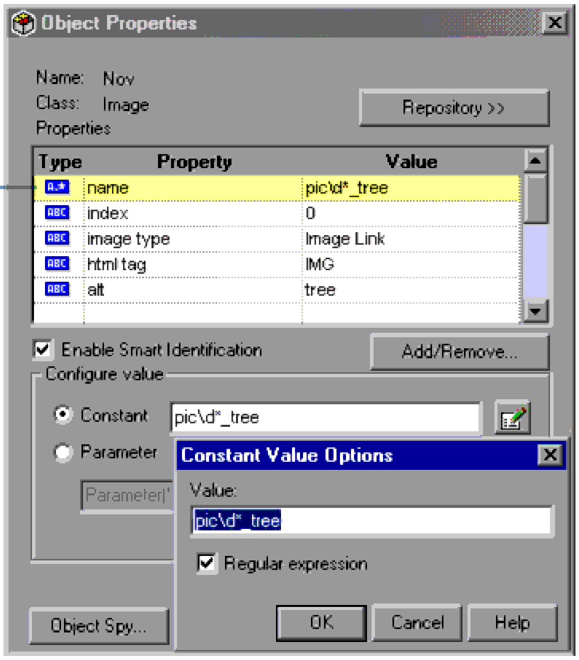
If certain object property values in your Web site or application are generated dynamically for all objects of a given type, you can create a property pattern configuration file that instructs UFT to automatically record the object property string as a regular expression.

> **Note:** The property pattern configuration file applies only to constant **identification properties** of type **string**. It does not convert the expected values of checkpoints (including automatic checkpoints) to regular expressions.

For example, suppose a Web site displays a number of thumbnail photographs and clipart images as the results of a search, with links to page containing a larger version of each image. The **name** property for each image is generated dynamically, so the name of the image is different each time you run your test, but you want to identify the image using the **name** property. Photograph names start with pic followed by a generated multi-digit code, followed by an underscore and a descriptive name of the image. Clipart images start with img followed by a similar pattern. For example, on a given test run, one image name might be: pic198362839_tree, and another image on the page might be named img2984730_apple.

You can create a property pattern definition for the **name** property of the Image object, so that the **name** property for all images is recorded with a regular expression that learns the property value as the first three literal characters, followed by any number of digits, and then all remaining literal characters. After recording, the **name** property value in the Object Properties dialog is automatically displayed with the appropriate regular expression:

The A.* icon indicates that the value is a regular expression



Object Properties

Name: Nov
Class: Image
Properties

Type	Property	Value
A.*	name	pic\d*_tree
ABC	index	0
ABC	image type	Image Link
ABC	html tag	IMG
ABC	alt	tree

☑ Enable Smart Identification

Configure value

● Constant: pic\d*_tree

○ Parameter: Parameter("

Constant Value Options

Value: pic\d*_tree

☑ Regular expression

You must perform the following three steps to use the property pattern option:

- activate the property pattern option in the registry
- create a property pattern configuration file
- select a configuration file

> **Note:** The modified registry and the property pattern configuration file must be set up on the computer on which you want to record objects using property patterns. However, once the test is recorded, you can run the test on any UFT computer.

Activating and Disabling the Property Pattern Option

If you want to use a property pattern configuration file, you must first activate the **UsePropertyPattern** option in the registry. If you no longer want to use the property pattern configuration file, you must disable the **UsePropertyPattern** option.

To activate or disable the UsePropertyPattern option:

1. Select **Run** from the **Start** menu. The Run dialog box opens.
2. Type regedit and click **OK**. The Registry Editor opens.
3. Double-click the **UsePropertyPattern** option from: *HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest QuickTest Professional\MicTest\AdditionalObjects\PropertyPatternObj\Settings.* The Edit DWORD Value dialog box opens.
4. To activate the Property Pattern option, enter 1 in the **Value Data** box.

   To disable the property pattern option, enter 0 in the **Value Data** box.

Creating the Property Pattern Configuration File

The property pattern configuration file is an XML file. The file must be saved with an *.xml extension. You can save the file in any folder.

The format of the Property Pattern configuration file is described below. Text in **bold** should be typed literally. Text in *italics* should be replaced with the appropriate information as described in the table below the format description.

You can create your property pattern configuration file in any text editor, but you should open the file in an application with an XML parser (such as Microsoft Internet Explorer) to confirm that the syntax is correct.

<XML>

<Object Name="*ObjectClass***">**

<Property Name="*PropertyName***">**

<Pattern ToFind="*SearchExpression***"**

ToReplace="*ReplaceExpression***">**

</Pattern>

...

</Object>

...

</XML>

Element	Description
<XML>..</XML>	**The XML element begins and ends the XML file.**
<Object>..</Object>	**The Object element defines the UFT test object class for which you want to define a pattern. Each element defines one object. You can create as many <Object>..</Object> elements as necessary.**
Name	**The Name attribute specifies the name of the test object class.**
ObjectClass	**The name of the UFT test object class for which you want to define a pattern, or "**Any Object**".** **Entering "**Any Object**" as the object class enables you to define patterns that UFT applies to any object you record with a property value matching the defined pattern if, and only if, no other object-specific pattern in the configuration file applies. In other words, a pattern defined for a specific object class overrides the definitions in the "**Any Object**" section of the file.**
<Property>..</Property>	**The Property element defines the UFT identification property for which you want to define a pattern. Each Property element defines one object property. You can create as many <Property>..</Property>** **elements as necessary for each object.**
Name	**The Name attribute specifies the name of the identification property.**
PropertyName	**The name of the UFT identification property for which you want to define a pattern.**

Element	Description
<Pattern>..</Pattern>	**The Pattern element defines the property pattern to find, and the regular expression with which it is replaced. Each element defines one pattern to find and replace. You can create as many <Pattern>..</Pattern>** **elements as necessary for each object property.**
ToFind	**The ToFind attribute specifies the expression to find.**
SearchExpression	**The expression to find. Use parentheses to divide the expression into parts.** **For example, if you want to find values containing 3 characters followed by several digits, followed by an underscore and several characters, enter: (\w{3})(\d*)(_\w*).** **Use standard regular expression syntax to define your search expression.**
ToReplace	**The ToReplace attribute defines the regular expression that replaces all occurrences of the pattern specified in the ToFind attribute.**
ReplaceExpression	**The expression that replaces the text of the specified pattern. If you want to retain a part of the text that was found, specify the part using $#, where # is the number of the part.** **For example, "$1.*" means—replace the text found with the contents found for the first part of the *SearchExpression* followed by ".*".** **So, if the text** pic12345_animal **was found using the SearchExpression—**(\w{3})(\d*)(_\w*)**, then the ReplaceExpression—**"$1\d*$3" **would replace the text with the regular expression—pic\d*_animal.**

Sample Property Pattern Configuration File

The following configuration file defines property patterns for Image objects as well as for "Any Object".

<XML>

<Object Name="Any Object">

<Property Name="Name">

<Pattern ToFind="(\w{5}-\w{2})(-\w{6}-)(\d*)"

ToReplace=".*$3"></Pattern>

<Pattern ToFind="(\w{5}-)(\d*)"

```
ToReplace=".*$2">
</Pattern>
</Property>
</Object>
<Object Name="Image">
<Property Name="Alt">
<Pattern ToFind="(\w{5}\.)(\w{4})(\d*)"
ToReplace=".*$3"></Pattern>
<Pattern ToFind="(\w{5}-)(\d*)"
ToReplace=".*$2">
</Pattern>
</Property>
</Object>
</XML>
```

Assume the image below was part of your Web site:

```
<IMG html id = 1 alt=Jenny.Foxx567 name=jfoxx height=29 src2.gif" width=65>
```

The property pattern configuration file above replaces the Alt property of the image to the following regular expression:

```
Jenny\..*567
```

Selecting the Property Pattern Configuration File

After you have created a property pattern configuration file, you must select the file in the PropertyPatternsFile option in the registry.

To select the Property Pattern file:

1. Select **Run** from the **Start** menu. The Run dialog box opens.

2. Type regedit and click **OK**. The Registry Editor opens.

3. Double-click the **UsePropertyPattern** option from: *HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest QuickTest Professional\MicTest\AdditionalObjects\PropertyPatternObj\Settings.* The Edit DWORD Value dialog box opens.

4. Enter **1** and click OK.

5. Double-click the **PropertyPatternFile** option from: *HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest QuickTestProfessional\MicTest\AdditionalObjects\PropertyPatternObj\Settings.* The Edit String dialog box opens.

6. Enter the full path for your property pattern configuration file and click **OK**.

Normalizing Property Values Before Applying Property Patterns

If the object property value to which you need to apply a property pattern may contain a special character, and you want those characters to be treated literally, you can use the **NormalizeValue** attribute of the optional **Settings** element in your property pattern **.xml** file to instruct UFT to insert a backslash before all special characters in the property value before applying the property pattern.

For example, if an image source file might be named: pic198362839_tree.gif, and you want to replace this value with pic\$\d*_tree\$\.gif, you can can set the **NormalizeValue** attribute to "1". This way, the dollar signs and dot (.) before the extension are automatically prefixed with backslashes before the rest of the property pattern file converts the digits in the filename with the \d*.

To add the **NormalizeValue** attribute to your property pattern file, add the Settings element to the top of your **.xml** file:

```
<XML>

<Settings NormalizeValue= "1"/>

<Object Name=.....
```

Part 8: API Testing Design

Chapter 54: API Test Creation Overview

Relevant for: API testing only

This chapter includes:

Concepts

UFT API Testing Overview

Relevant for: API testing only

API Testing is how UFT enables you to test your non-GUI applications to see how your application's API processes performs. Using UFT API testing, you create a test that represents the activities that your application performs (even using actual and output data from your application), and use checkpoints to assess the success or failure of the test.

You design an API test by dragging activities from the Toolbox pane onto the canvas. The Toolbox pane includes a collection of standard activities and technology-specific activities such as HTTP, Java, JMS, IBM WebSphere MQ, and SAP. You can add additional activities to the Toolbox pane by importing WSDLs and WADLs, creating REST Services, or using services from the file system or ALM. You can also create new custom activities using the built-in Activity Wizard as described in "API Testing Extensibility" on page 2022.

This section includes:

Testing Application Processes with API Testing

Relevant for: API testing only

To test your application's processes and services, you create test steps modeled on API activities and check the results to determine whether your application is performing as expected..

For example, if your Web-based application requires you to log in to the application, you can create test steps to check that the application responds correctly to a HTTP Request, that the application connects to the database containing user data, and so on. You can test these processes by using the activities provided with UFT, or you can import Web Services and REST Services to create activities that represent your application’s services

You create a basic test by dragging an activity from the Toolbox pane to the canvas, assigning the activity input and output data or parameters, and then setting checkpoints to verify the response.

For more details, see "How to Create an API Test" on page 1598.

UFT provides a number of standard activities that test common application processes, including:

- **Flow Control** activities, such as **Wait**, **Break**, and **Conditional** steps. These activities enable you to customize your test to match any special application workflows.
- **String Manipulation** activities, such as **Concatenate Strings** and **Replace String**.

- **File system** activities, which enable you to test your application's interaction with the file system.
- **Database** activities, which enable you to test your application's ability to connect and communicate with a database.
- **FTP** activities, which enable you to test your application's ability to perform FTP-related procedures.
- **Network** activities, such as **HTTP Request** and **SOAP Request**, which enable you to test your application's connection with a network or web-based server.
- **JSON** and **XML** activities, which enable you to test your application's ability to convert XML and JSON data to text and vice versa.
- **Math** and **Date/Time** activities, which enable you to perform math operations or perform tasks using the system date and time.
- Other **Miscellaneous** activities, including **Custom Code** activities, **Run** and **End** program activities, and a **Report** activity.

In addition, there are various activities directed at testing application processes using specific technologies, including:

- A **Call Java Class** activity, which enables you to test Java-based processes that run in your application
- **JMS** (Java Message Service) activities, which enable you to test your application's ability to communicate, publish, browse, receive, and check messages from a JMS queue.
- **IBM WebSphere MQ** activities, which enable you to test your application's ability to communicate with, publish, browse, receive, and check messages from the IBM WebSphere MQ queue or topic.
- **SAP** activities, which enable you to test your application's ability to communicate with an SAP server using IDOCS and RFCs.
- **Load Testing** activities, which enable you to add steps for your test to be run as a load test in HP LoadRunner.
- **HP Automated Testing Tools** activities, which enable you to call a GUI test or action, API test or action, or Virtual User Generator script from UFT, QuickTest Professional, Service Test, or LoadRunner to use as part of your test.

For details on these activities, see "Standard Activities" on page 1611.

If you need additional functionality beyond the standard activities, you can import or create custom activities, including:

- **Web Service** activities imported from a WSDL file. For details, see "How to Import a WSDL-Based Web Service" on page 1744.
- **Web Application** activities imported from a WADL file. For details, see "How to Import a Web Application Service" on page 1754,
- **REST Service** activities created in UFT using the Add REST Service dialog box. For details, see "How to a Create a REST Service" on page 1746.

Any of these activities can be dragged to the canvas and used in a test flow.

Extensibility in API Testing - Overview

Relevant for: API testing only

For greater control over test steps, you can use custom code activities that seamlessly integrate with UFT. Using customized code, you can also customize the behavior of existing activities using event handlers. For details on creating and using custom code with your test, see "Writing Event Handlers for API Test Steps" on page 1935.

If the existing test activities are not suitable to test your application, you can also create custom test activities with the UFT Activity Wizard (**Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Activity Wizard**). The Activity Wizard enables you to specify the activity type and properties. It then exports the activity to the Toolbox pane for use in future testing sessions.

For details on adding new activities and custom code, see "API Testing Extensibility" on page 2022.

> **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

API Test Creation Methodology

Relevant for: API testing only

Using the UFT's test creation methodology, you can create tests for your applications earlier in the application development process. This test design methodology separates the test step *design* from the test step *infrastructure* design. As a result, you can develop test steps earlier and maintain these steps with only minor updates, even when the application or testing needs change significantly.This process is also especially useful because it offers a clear division of test design tasks. A few people maintain the resource framework while other users design and maintain test steps. Additionally, after the basic infrastructure is in place, both types of users can do their jobs simultaneously.

The information in the sections listed below provides an overview of the main steps involved in creating an API test. For a task description of this process, see "How to Create an API Test" on page 1598. For a use-case scenario description of this process, see "How to Create an API Test - Use-Case Scenario" on page 1603.

Analyzing Your Application

Relevant for: API testing only

The first step in creating a test is to analyze your application to determine your testing needs.

What technologies do you want to test?

From the perspective of UFT, your application consists of numerous business processes that run functions created to perform certain tasks. UFT provides support for these business processes using standard activities. Furthermore, UFT enables you to import Web Services activities, WADL activities, and user-created activities as well as create your REST Services activities.

Inside UFT, you can use the standard activities or import your own custom activities by importing your WSDL or WADL files or creating your REST service methods using the REST Service designer. These activities are displayed in the Toolbox pane, enabling you to select from a wide range of potential activities to use in your test. If you need to create additional activities, you can use the Activity Wizard to represent your application's activities and then import them into UFT.

What information do you need to know to test the business processes in your application?

For each of your application processes, you will need to know the properties of these processes, including the input and output properties passed from the application, any input or output parameters, and server information (for Web-based steps). If you are working with network-based or Web-based activities, (such as WSDL, REST, and WADL), you need to know the URL properties for your application, as well as additional information such as proxies and security certificates required

These properties are entered into the Properties pane for each test step, enabling UFT to effectively test the functionality of the application using the application's actual properties and parameters.

For details on the Properties pane and the various tabs used for test activities, see "Properties Pane User Interface (API Testing) " on page 402.

What actions do you need to create?

If you find that the application processes are consistently using a series of repeated activities, you can create test actions that contain the repeated steps. After you create the action, instead of entering the repeated steps and entering their properties multiple times when creating your test, the test can call the action and run the test steps contained in the action.

> **Example**
>
> Your application that enables customers to book a flight online contains various business processes, including logging on to the application, selecting a flight, and creating a customer order. As part of these processes, your application has to check the application database for the information and return this data to the application's user interface. For each of the application processes, you could create an action that contains a database connection step, a step to retrieve the necessary data, and a step to return the data to the application. Each time you test a different application process, you can call this action as a precursor to the other test steps.

By creating actions for repeated test steps, you also make your test design more efficient.

You can also call an action from an external test, thereby making these test steps reusable across multiple tests.

> **Tip:** As you plan your tests and actions, keep in mind that short tests and actions that check specific application processes are often better than long tests that perform several tasks, as they are easier to reuse and maintain over time.

Configuring UFT According to Your Testing Needs

Relevant for: API testing only

Before creating your tests, you need to configure UFT to suit your needs. This involves one or more of the following:

Defining your global testing preferences

You need to specify configuration settings that affect how you create and run tests in general -- these settings are not test-specific.

Global testing preferences in the Options dialog box (**Tools > Options**). For details, see "UFT Global Options" on page 522.

Configuring test-specific preferences and properties

You can also set a number of test-specific properties:

- **General test properties:** Using the General tab of the Properties pane, you set the global general test properties. For details, see "General Tab (Properties Pane - API Testing)" on page 413.

- **User profiles:** You can create user profiles for each test to enable multiple users to define different variable values for the same test. For details see, "How to Define API Test Properties or User/System Variables" on page 141.

- **Test specific properties/parameters:** You can set custom input and output property parameters that can be used by any test step by selecting the **Start** or **End** of the test.

Configuring the UFT IDE to suit your testing preferences

You can open, close, and move any of the UFT panes to suit your needs when creating tests. This enables you to easily access any needed panes, such as the canvas, the Solution Explorer pane, the Toolbox, Properties pane, or the Data pane easily. For details, see "UFT Window Layout" on page 182.

Building and Enhancing Your API Tests

Relevant for: API testing only

Before creating your test steps, you may need to import your Web Service (WSDL) and Web Application methods or create your REST Service methods. For details, see "Local Activities" on page 1733.

After creating a test, you modify it by dragging activities from the Toolbox pane to the canvas and modifying the step properties in the Properties pane. You can also use a variety of options to further enhance your tests:

Checkpoints

For many test activities, you can define checkpoints. In an API test, a **checkpoint** is set as a property of a test step. Using checkpoint properties enables you to determine whether or not your application is functioning correctly.

For details on the different checkpoints properties available for test activities, see "Standard Activities" on page 1611.

Parameterization

When you test your application, you may want to check how it performs the same processes with different data. You can do this by replacing fixed values from an external data source during your run session. This is called **parameterizing** your test. You can supply data from an Excel data table, a local data table, a database, or an XML file. For more details, see "Data Usage in API Tests" on page 1800. In addition, you can data drive all the steps in your test, using Excel or XML data sources.

Linking Steps

You can retrieve values from your test steps and use them as input or output properties/parameters for other test steps. This enables you to use data or values retrieved from application processes (in much the same manner your application does) in other parts of your test. You can also create custom link expressions combining values from a data source, other test steps, or test variables. For details, see "Linking Property Values to a Data Source" on page 1803.

Actions

You can add actions to your test to combine steps that are run multiple times in the course of a test. By combining these repeated steps in a single action and calling the action later in a test, you make test creation more efficient.

Custom Code

You can add custom code activities using C# code to be run as a test step. You can also add event handlers to any of your test steps, enabling you to customize the behavior of the test step before, during, and after running the step, as well as coding special checkpoint events for each test step. Each event handler or custom code has autocomplete options available to help you write your code.

When creating custom code, you may do the following:

- Determine whether you need to create additional event handlers for a particular test step or create a new user-defined code document
- Determine what functions are needed
- Add custom code documents to your test. For details, see the "Add Reference Dialog Box" on page 485.
- Debug your custom code. For details, see "How to Debug Your Test, Component, Function Library, or User Code File" on page 683.

For details on using custom code, see "Writing Event Handlers for API Test Steps" on page 1935.

Running and Troubleshooting Your API Tests

Relevant for: API testing only

When your tests are complete and ready to run, you run them, view the run results, and troubleshoot your tests, as needed.

After your test, view the run results. Expand the nodes in the Run Results Viewer to see where steps succeeded and failed and to try to understand why. For details, see the *HP Run Results Viewer User Guide*.

For details on running your tests, see "Running Tests and Components" on page 634

For details on debugging your tests, see "Debugging Tests and Components" on page 674.

Automated Testing Tool Integration

Relevant for: API testing only

UFT API testing integrates with the following other HP products:

- **HP Application Lifecycle Management (ALM).** You can save tests, business component, and test resources on ALM, enabling multiple users to store and access a shared repository of tests and test resources. You can also use ALM's defect tracking abilities to record and manage application defects when running your tests. For details on UFT-API testing integration with ALM, see "ALM Integration" on page 707. For details about ALM, see the *HP Application Lifecycle Management User Guide*.

- **HP Service Test Management.** Service Test Management integrates with ALM and UFT API Testing to enable you to better use and organize your Web services. For more details, see the *Service Test Management User Guide*.

- **HP Service Virtualization.** In order to mimic services that your application may use, UFT integrates with HP Service Virtualization. After creating your service models in Service Virtualization, you then run them in tandem with a UFT API test. For more details on using Service Virtualization in UFT, see "Running Tests with Virtualized Services - Overview" on page 665. For more details on HP Service Virtualization, see the *Service Virtualization User Guide*.

- In addition, by using **HP Automated Testing Tools** activities you can integrate with additional HP functional testing products by creating test steps that call a UFT GUI test or action, UFT API test or action, or a LoadRunner script. These tests or scripts are created in the original application and call from within your UFT API test flow.

  For task details, see "How to Call External Tests or Actions" on page 1639.

Tasks

How to Create an API Test

Relevant for: API testing only

This task describes the workflow and methodology to follow when you create and build a test.

This task includes the following steps:

- "Prerequisite - Analyze your application" below
- "Prerequisite - Configure UFT according to your testing needs" below
- "Prerequisite - Prepare the service references (optional)" on the next page
- "Build your test structure" on the next page
- "Enhance your test steps" on page 1600
- "Results" on page 1602

Prerequisite - Analyze your application

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

- **Determine the functionality you want to test.** To do this, you consider the various activities that the application performs. What business processes run? What activities are most relevant for the business processes that you want to test?

- **Identify any processes that run repeatedly.** Plan to create actions in your test for such processes.

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

Prerequisite - Configure UFT according to your testing needs

This can include:

- Setting up your **global testing preferences**.

  For details, see "UFT Global Options" on page 522.

- Setting up **test-specific preferences**, including global properties, test input and output properties and parameters, or user profiles and variables.

  For details, see "How to Define API Test Properties or User/System Variables" on page 141.

- Configuring your **run session preferences**. For details, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 526.

Prerequisite - Prepare the service references (optional)

- **Import or build the set of resources** to be used by your tests, including:

  - **Importing WSDL or WADL files**, which define the methods used for a Web service or Web application.

    For details, see "How to Import a WSDL-Based Web Service" on page 1744 or "How to Import a Web Application Service" on page 1754.

  - **Creating REST service resources and methods** if your application is based upon REST services.

    For details, see "How to a Create a REST Service" on page 1746.

  - **Importing .NET assemblies** referenced by your test.

    For details, see "How to Import and Create a .NET Assembly API Test Step" on page 1760.

  - Adding any **additional reference files** that will be referenced by your test.

    For details, see "Add Reference Dialog Box" on page 485.

Note: Skip this step when you will be using only the built-in operations. These activities are available in the Toolbox pane under the Standard Activities section.

Build your test structure

To build your basic test structure, do the following:

1. **Create additional test flow steps- optional**

   Expand the Toolbox pane nodes and drag **Flow Control** activities onto the canvas:

   - **Loop.** Enables you to add another loop (the **Test Flow** loop is always part of a test, and cannot be removed). You specify the loop behavior in the loop's input properties.

   - **Condition**. Enables you to define conditional branches.

   - **Sleep.** Indicates a time delay in milliseconds.

2. **Add activities to the test to create test steps**

   Expand the nodes of the Toolbox pane and drag activities into the **Text Flow** or a **Loop** box within the canvas to create test steps. If you added a **Condition** step, drag activities into the condition branches. For a list of the available activities, see the "Standard Activity Types" on page 1650.

3. **Provide the step properties**

   Enter the input, output, and checkpoint properties (as needed) for each activity. For details on the input, output, and checkpoint properties available for each activity, see "Standard Activities" on page 1611.

   > **Note:** If you have a number of activities/steps that are repeated often in your test, consider creating an action and adding the steps to this action. After creating the action, you can call the action each time you need to repeat these steps in your test instead of adding the activities and setting the activity's properties repeatedly.

Enhance your test steps

To enhance your test steps, do any of the following:

- **Define data sources for the test**

  For details, see "How to Assign Data to API Test/Component Steps" on page 1819 or "How to Data Drive an API Test Step" on page 1827.

- **Create a Custom Code activity**

  a. Select the **Custom Code** activity from the **Miscellaneous** category and drag it into a loop.

  b. Click the Input/Checkpoints tab in the Properties pane.

  c. Click **Add Properties** and create the required input and output properties.

  d. Open the **Events** tab in the Properties pane.

  e. Double-click the **Handler** column of the **ExecuteEvent** row. UFT opens a new tab TestUserCode.cs.

  f. Locate the **Todo** section and enter your custom code. Follow the sample code in the comments and use autocompletion to write your code.

  g. Click **File > Save All** to save the custom code and the test.

  For details and examples, see "Writing Event Handlers for API Test Steps" on page 1935.

- **Add input attachments to the test - optional**

  For Web Service activities that support input attachments, add an input attachment.

a. Select the Web Service call step in the canvas and open the **Attachments** 📎 tab in the Properties pane.

b. In the upper pane, select an attachment Type: **DIME** or **MIME**.

c. Click in the **Attachments** row and click the **Add** button ✚ to add an array element.

d. Select a file as the **Origin** of the attachment using the **Browse** button ⋯ .

e. Select a **Content Type**. Specify a **Content ID** or keep the default value, **Auto**.

- **Add/validate an output attachment- optional**

  a. Select the Web Service call step in the canvas and open the **Attachments** 📎 view in the Properties pane.

  b. Click in the **Attachments** row in the Checkpoints pane, and click **Add** ✚ to add an array element (it may be necessary to expand the column).

  c. Select the check box adjacent to each item that you want to validate. Specify values for the elements being validated: **Content Type** and/or **Content ID**.

  d. To validate content, click the **Calculate the file checksum** ⋯ button icon in the **Content** row. UFT calculates the specified file's checksum using the MD5 Hash function.

  For user interface details, see "Attachments Tab (Properties Pane - API Testing)" on page 403. You can also check for received attachments in the test's folder, stored as files with a .bin extension.

- **Add an event handler - optional**

  For any activities, you can define default event handlers for checkpoints, and before and after step executions.

  To add an event handler for a step:

  a. Select a step in the canvas and open the Events tab ⚡ in the Properties pane.

  b. In the row containing the event execution point you want (before, after,etc.), select **Create a default handler**.

  c. Edit the code in the TestUserCode.cs tab. Locate the **Todo** section and add your custom code. Follow the sample code in the comments and use statement completion to create an expression.

d. To access the properties of an activity, cast it before the activity name. For example:

```
ConcatenateStringsActivity cat = args.Activity as ConcatenateStringsActivity;
args.Checkpoint.Assert.Equals(cat.Prefix+cat.Suffix, cat.Result);
```

e. Click **File > Save All** to save the TestUserCode.cs file and the test.

For details and examples, see "Writing Event Handlers for API Test Steps" on page 1935.

> **Tip:** To ignore an event handler for a step, select a handler and press DELETE to clear the field in the Events tab. To completely remove the event handler, you must also delete the code manually in the TestUserCode.cs tab.

Results

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

- **Run your test to check your application.**

  The test starts running from the Start step in the canvas and stops at the end of the test. While running, UFT performs each step in your test, including any checkpoints.

  If you parameterized the test using data from a data source stored in the Data pane, UFT repeats the test (or test flow loop, if needed) using the data as defined in the Input tab for the **Test Flow** or test flow steps.

- **Run your test to debug it.**

  > **Note:** Before running a debugging session, make sure to enable debugging capabilities by selecting the **Run test in debugging mode** option in the "General Pane (Options Dialog Box > API Testing Tab)" (described on page 565).

  For general details on debugging, see "Debugging Tests and Components" on page 674. For details on the available debugging panes, see "Debug Panes" on page 271.

- **Run an individual step.**

  Select the step in the canvas and choose **Run Step** from the context menu to run the step with its property values. Note the results in the **Run Step Results** pane, in the lower section of the main window. If something needs to be modified, do so at this point. For details, see "Run Step Results Pane" on page 454.

  > **Note:** The Run Step command, for steps whose properties are assigned an XML data

> source, does not retrieve the latest data shown in the user interface.
>
> **Workaround**: Save the test before activating the Run Step command.

How to Create an API Test - Use-Case Scenario

Relevant for: API testing only

Creating a test is comprised of several stages. This section walks you through the stages you might perform when preparing a test for the Flight API application.

This scenario consists of the following steps:

- "Analyze the Flight API application" below
- "Create or import your test resources" on page 1605
- "Create your test steps" on page 1606
- "Enhance your test steps" on page 1609
- "Run the test" on page 1610

1. **Analyze the Flight API application**

   When analyzing the application to determine what application processes you may want to test, you can consider the existing business processes that run in the application.

   The business processes that should be tested for the Flight API application include:

   - Connecting to the user database and confirming login information
   - Retrieving the list of flights
   - Retrieving a flight order based on user input
   - Creating a flight order
   - Updating a flight order
   - Deleting a flight order
   - Deleting all flight orders
   - Logging a user out of the application

   Although the first and last items above have not yet been implemented in the Flight API application that you want to test, it is important to take them into account in the planning stage.

Now that you have determined the primary application processes, you should analyze each one to determine the breakdown of these application processes into smaller, testable parts.

A logical breakdown of the above business processes could be:

- **Connecting to the user database and confirming login information**
    - Sending a request to connect to the database
    - Receiving confirmation or failure of database connection
    - Checking the customer login details database based on user input
    - Returning authentication results (permission or rejection) for user input
    - Return results to application user interface
- **Retrieving a list of flights**
    - Connecting to the flight information database
    - Receiving confirmation or failure of database connection
    - Searching the flight information database for all available flights
    - Returning the search results
- **Retrieving a flight order based on user input**
    - Connecting to the flight information database
    - Receiving confirmation or failure of the database connection
    - Searching the flight information database using the user-specified criteria
    - Returning the search results
- **Creating a flight order**
    - Connecting to the flight database for the specific flight based on user input
    - Receiving confirmation or failure of the database connection
    - Reserving a place on the selected flight in the database
    - Returning the order confirmation
- **Updating a flight order**

    - Connecting to the flight database and finding the selected flight based on user input
    - Receiving confirmation or failure of the database connection
    - Retrieving the flight order details
    - Updating the flight information in the database based on user input
    - Returning the flight order update confirmation

   - **Deleting a flight order**

     - Connecting to the flight database and finding the selected flight based on user input
     - Receiving confirmation or failure of the database connection
     - Retrieving the flight order details
     - Deleting the flight order from the database
     - Returning confirmation of the flight order deletion

   By comparing the sub-steps in each of the business processes, you can see what specific steps you need to design in your test. In addition, you can also see the steps that are repeated and can be combined in a reusable action that can be called in different parts of your test.

2. **Create or import your test resources**

   Some of your application processes for the Flight API application require custom activities not included in the standard collection of API activities provided in the Toolbox pane. For these activities, you must import or create the activities into your tests.

   At this stage we can import the following Web services methods:

   - CreateFlightOrder
   - GetFlights
   - GetFlightOrders
   - UpdateFlightOrder
   - DeleteFlightOrder
   - DeleteAllFlightOrders

   For details on how to import Web Service methods, see "How to Import a WSDL-Based Web Service" on page 1744.

You can also create the following REST Service resources and methods:

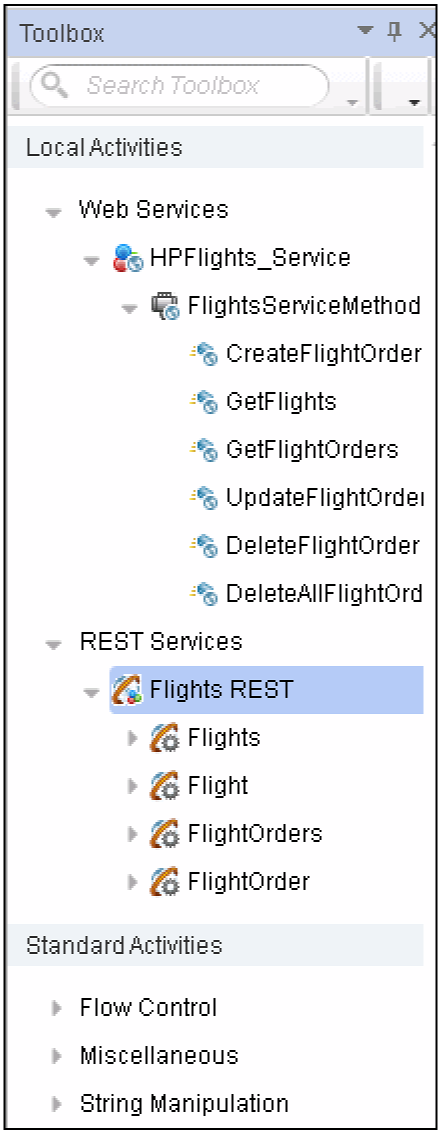
- Flights Get
- Flight Get
- FlightOrders Get
- FlightOrders ReserveOrder
- FlightOrder Get
- FlightOrder Update
- FlightOrder Delete
- FlightOrder DeleteAll

For details on how to create REST Service methods, see "How to a Create a REST Service" on page 1746.

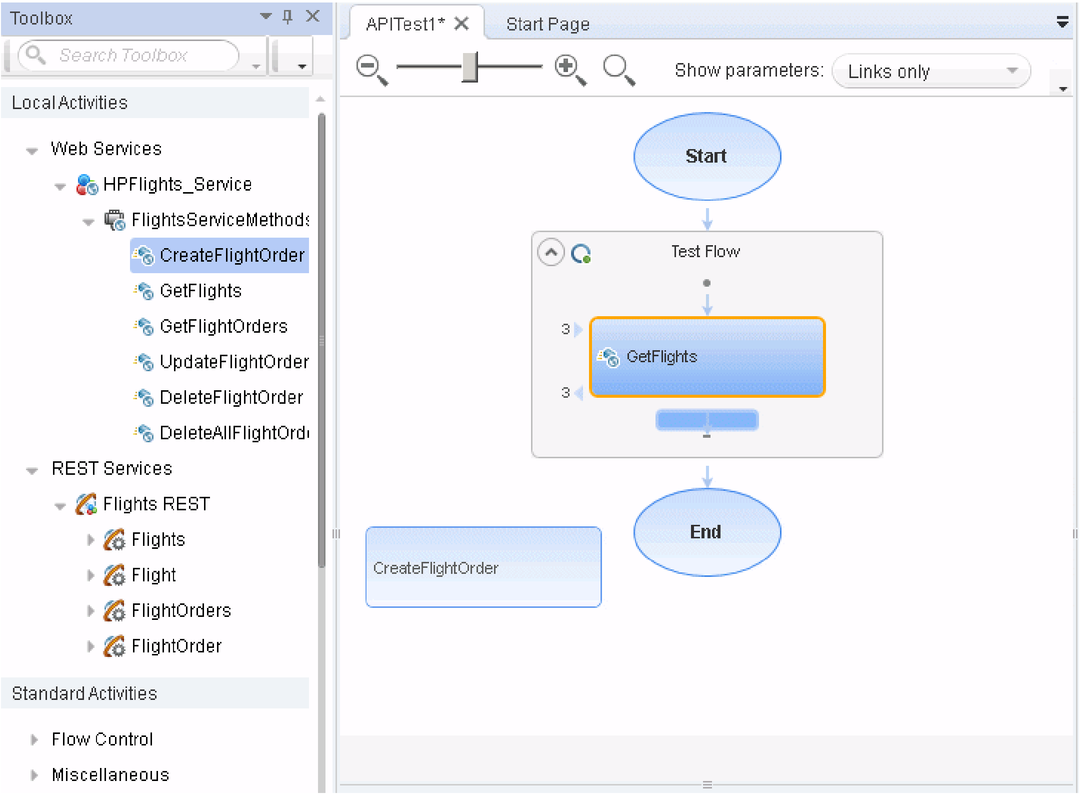
3. **Create your test steps**

   Now that you have planned and prepared all of the required resources for your test, you are ready to create test steps that represent the steps a real application would perform on the Flight API application.

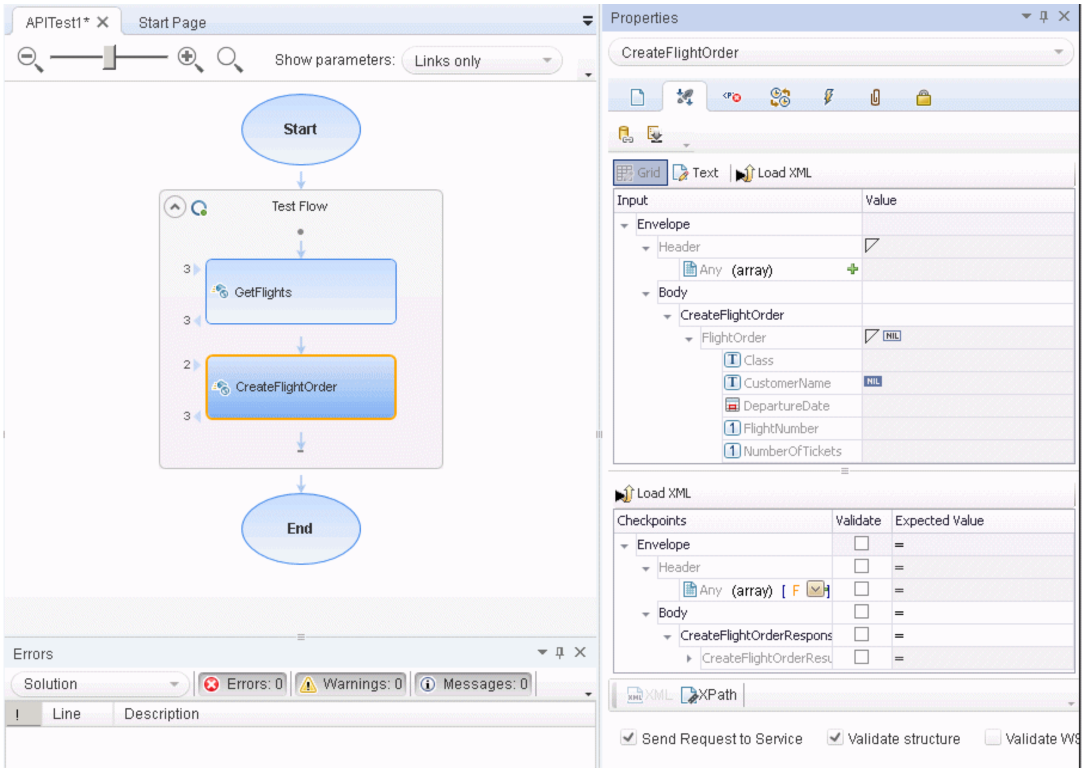
The activities that are available for your test steps are stored in the Toolbox pane. This includes custom methods that were imported or created for your test:



To create test steps, you select activities from the Toolbox pane and either drag them to the canvas or double-click them to add them to the canvas:



You then set input, output, or checkpoint property values for the activities to mimic your application:



If you have steps that appeared multiple times, you can create an action which combines the repeated steps and call this action instead of the repeated steps.

For task details on creating test steps, see "How to Create an API Test" on page 1598.

4. **Enhance your test steps**

Once you have put the appropriate steps for the Flight API application in the canvas, you can add additional enhancements for these test steps:

- You can set checkpoint properties for a test step, providing the expected output values for each of the methods.
- You can link the test steps to a data source, such as an Excel file containing different values for the step input properties. This enables you see how the test steps run with different input values.
- You can link one step to another. For example, you can link the GetFlights Web Service method to the CreateFlightOrder method to pass the flight information between the two methods.

- You can add additional event handlers to enhance the step function before the test step runs, during the running of the test step, and after the step's execution.

5. **Run the test**

   After you add your test steps, and provide the input, output, or checkpoint property values, you run your test and observe the results. Expand the Run Results tree and observe the results and response for each of the test steps.

   If you want to test each individual step after entering its properties, you can right-click the step name in the canvas and select Run Step. UFT runs only that step and displays the step results in the Run Step Results pane.

Chapter 55: Standard Activities

Relevant for: API testing only

This chapter includes:

Concepts

Activity Overview

Relevant for: API testing only

You create a test by double-clicking or dragging **activities** from the Toolbox pane into a canvas. The Toolbox pane provides a collection of built-in standard activities for functional testing in areas such as file and string manipulation, and messaging through HTTP, FTP, and JMS.

The activities are divided into several categories:

- **Standard Activities.** This category includes the built-in activities, such as **String Manipulation**, **Database**, **Network**, **File System**. For a complete list of the standard activities, see "Standard Activity Types" on page 1650.

- **Local Activities.** This category includes the activities that are stored as part of the test or business component. These can be imported services such as Web or REST service operations, and .NET assemblies. For details, see "Local Activities" on page 1733.

- **File System Activities.** This category includes activities that reside in the file system on either local or network drives. These are Local activities that you moved into the file system repository that can be shared between tests.

- **ALM Activities.** This category includes the activities that reside in the ALM repository. These are Local activities that you moved into the ALM repository that can be shared between tests. This category only appears when a connection to an ALM server is open.

You can also create new custom activities, using the extensibility API provided with the product. These will be stored under the Standard Activities. For details, see "API Testing Extensibility" on page 2022.

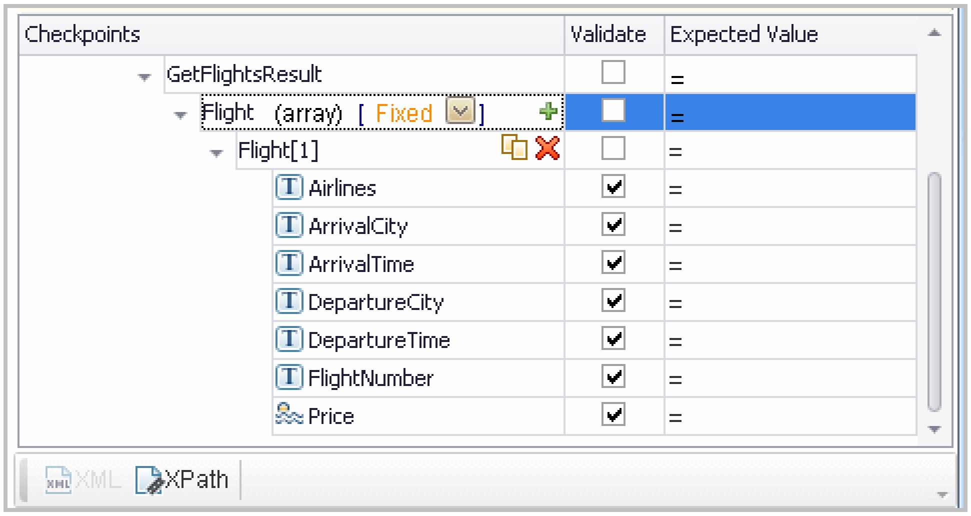
For general information about the API interface and creating tests, see "API Test Creation Overview" on page 1589.

Checkpoint Validation for Test Steps

Relevant for: API testing only

In functional testing, it is necessary to check the results of test steps to confirm that the activity performed its expected functionality. The response can contain several properties, each containing several data items. The **Checkpoints** section is a central point for defining the expected values of the properties.

Before replaying a test, you set the expected output property values. You can enter the values manually, link them to a data source, or load values that were captured during a prior replay. This is useful when you have many argument values—instead of manually entering values, you automatically load them.



Checkpoints	Validate	Expected Value
GetFlightsResult	☐	=
Flight (array) [Fixed]	☐	=
Flight[1]	☐	=
Airlines	☑	=
ArrivalCity	☑	=
ArrivalTime	☑	=
DepartureCity	☑	=
DepartureTime	☑	=
FlightNumber	☑	=
Price	☑	=

To include a checkpoint in the test, you select its check box. If you loaded replay values, you can select only those properties that you loaded.

For WSDL-based Web Services and SOAP Requests, UFT includes two built-in checkpoints for the purpose of validation. One checkpoint validates the XML structure and the other checks its compliance with WS-I.

Additional checkpoint settings let you trim the string, ignore case inconsistencies, and indicate whether to stop on failed checkpoints.

For details, see the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

After the test run, the Run Results Viewer displays the checkpoint status in its own sub-node. For details, see the *HP Run Results Viewer User Guide*.

This section also includes

XPath Checkpoints

Relevant for: API testing only

For steps with XML output properties, such as **Web Service** and **SOAP Request**, **String to XML**, and so forth, you can validate the test results against XPath expressions.

You can specify a fully qualified XPath expression, or you can instruct UFT to ignore the namespaces and prefixes during the test run. By ignoring the namespaces, you can use a simpler expression.

In the following example, to retrieve the contents of the second node, B, you would need to write an expression that also indicates the namespace, such as //*[local-name(.)='Node' and namespace-uri(.)='ns2'].

```
<Root>
  <Body>
      <Node xmlns="ns1">A </Node>
      <Node xmlns="ns2">B </Node>
  </Body>
</Root>
```

When working with simple XPath expressions, you can further simplify the XPath expression by selecting **Ignore namespaces**. In the above example, the expression //Node[2] is sufficient to evaluate the value B in the second node.

You can type in the XPath expressions manually, or use the shortcut menu to retrieve the simplified or fully qualified XPath.

UFT also enables you to evaluate XML with namespace prefixes. For example, if the XML contains the prefix definition xmlns:T="ns1", you can specify the prefix in the XPath expression: //T:*NodeName*. To evaluate namespace prefixes, disable the **Ignore namespaces** option.

XPath checkpoints can only be used when the XPath query returns a scalar value—not XML.

For task details, see "How to Set XPath Checkpoints for Test Steps" on page 1617.

Tasks

How to Set Array Checkpoints for Test Steps

Relevant for: API testing only

This task describes how to manually set checkpoints for elements of an array. For details about setting regular checkpoints, see "How to Run an API Test" on page 643.

For details about data driving array checkpoints, see "How to Data Drive Array Checkpoints" on page 1828.

This task includes the following steps:

- "Enable active content on your computer" below
- "Add a step with an array output" below
- "Select an array validation method" below
- "Add array elements" on the next page
- "Provide values" on the next page
- "Validate the element count - optional" on the next page
- "Set the number of iterations and run the test" on the next page
- "Open the Checkpoint report" on page 1617

1. ### Enable active content on your computer

   The Run Results Viewer shows array checkpoint results in an expandable tree. To enable this view, modify your browser settings as follows:

   a. In Internet Explorer, select **Tools > Options**.

   b. Select the **Advanced** tab.

   c. Enable the option **Allow active content to run in files on My Computer** in the **Security** section.

   d. Click **OK** and close the browser.

2. ### Add a step with an array output

   Add a test step with output properties in the form of an array.

3. ### Select an array validation method

Expand the drop down adjacent to the name of the parent array node. Select one of the following:

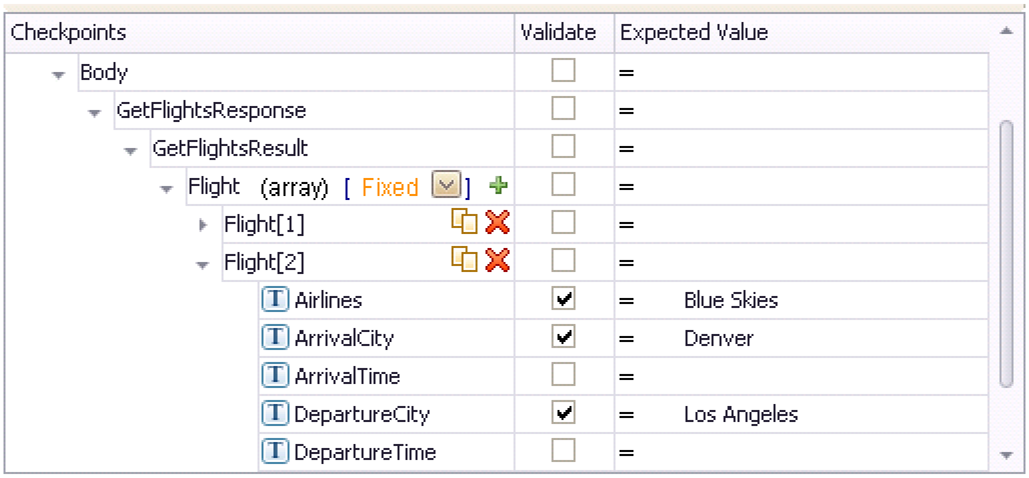
- **Fixed.** Checks that each of the returned array elements matches its corresponding array element in the **Checkpoints** pane. Each array is marked by an index number, as it checks the arrays by their index.
- **All.** Checks that all of the returned array elements match the array element in the **Checkpoints** pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked **>= 2** and the same property in another array element is set to **<=10**, the test run will check that all returned values are between 2 and 10.
- **Contains.** Checks that at least one of the returned array elements matches the value of the property in the **Checkpoints** pane. In this mode, arrays are not marked by an index number.

4. ## Add array elements

Use the plus button ✚ in the row of the parent node to add the desired number of array elements.

5. ## Provide values

Select the **Validate** box for each value that you want to validate and provide values for those properties.



Checkpoints	Validate	Expected Value
Body	☐	=
GetFlightsResponse	☐	=
GetFlightsResult	☐	=
Flight (array) [Fixed] ✚	☐	=
Flight[1]	☐	=
Flight[2]	☐	=
Airlines	☑	= Blue Skies
ArrivalCity	☑	= Denver
ArrivalTime	☐	=
DepartureCity	☑	= Los Angeles
DepartureTime	☐	=

6. ## Validate the element count - optional

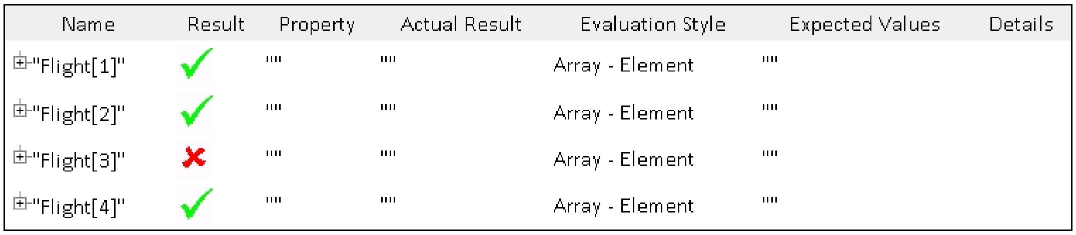
Click in the parent row of the array and enter a desired count number in the **Expected Value** column and the evaluation expression, such as **=**, **>**, and so forth. During the test run, the validation will check the number of returned array elements against this value.

7. ## Set the number of iterations and run the test

Click in the **Test Flow** or **Loop** box and open the Properties pane. Set the test flow properties, such as the number of iterations or loop type. Click the **Run** button and provide a results location. For details, see "How to Run an API Test" on page 643.

8. **Open the Checkpoint report**

   In the Run Result Viewer, expand the test results tree in the left pane. Select a Checkpoint node. In the **Captured Data** pane, click **View Report** in the Details column. In the report that opens, expand the checkpoint node for each of the iterations to view its actual and expected value.



Name	Result	Property	Actual Result	Evaluation Style	Expected Values	Details
⊞"Flight[1]"	✓	""	""	Array - Element	""	
⊞"Flight[2]"	✓	""	""	Array - Element	""	
⊞"Flight[3]"	✗	""	""	Array - Element	""	
⊞"Flight[4]"	✓	""	""	Array - Element	""	

How to Set XPath Checkpoints for Test Steps

Relevant for: API testing only

This task describes how to validate your test results against XPath expressions. For conceptual information, see "XPath Checkpoints" on page 1613.

This task includes the following steps:

- "Add a step with XML output" below
- "Set the namespace setting" below
- "Copy the XPath" on the next page
- "Add an XPath checkpoint" on the next page
- "Provide the XPath expression" on the next page
- "Set up the validation" on the next page
- "Run the test" on the next page

1. **Add a step with XML output**

   Add a test step with XML output, such as **String to XML**. Open the **Input/Checkpoints** tab . Paste a source string into the **Value** column or use the **Link to data source** button to link to a value.

2. **Set the namespace setting**

   Click the **XPath** tab, and indicate whether or not to ignore namespaces. By default, the **Ignore namespaces** option is enabled. If you plan to validate against a fully qualified expression, or if you need to specify a namespace prefix, disable the option. For details, see "<XPath checkpoint options>(available for steps with XML output properties)" on page 424.

3. ### Copy the XPath

   Copy an XPath expression to the clipboard. If you intend to enter the XPath expression manually, skip this step.

   - To retrieve a simple XPath, click in the **Value** column, and select **Copy XPath** from the shortcut menu.

   - To retrieve a complete qualified XPath, click in the **Value** column, and select **Copy Fully Qualified XPath** from the shortcut menu.

4. ### Add an XPath checkpoint

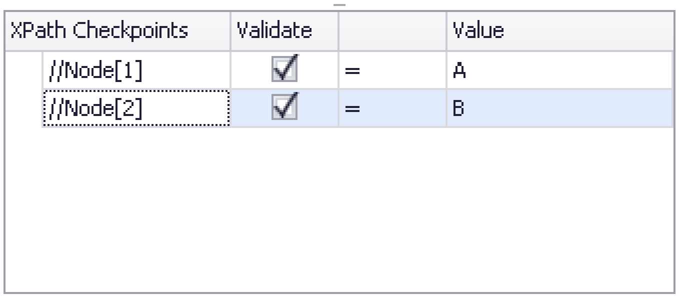
   Click the **XPath** tab in the Checkpoints section. Click the **Add** button ✚ to add a new XPath checkpoint.

5. ### Provide the XPath expression

   Type or paste the expression into the **XPath Checkpoints** column. You can also use the **Link to data source** button to link to a value.

6. ### Set up the validation

   Select the check box in the **Validate** column, select a comparison operator, and provide an expected value.



XPath Checkpoints	Validate		Value
//Node[1]	☑	=	A
//Node[2]	☑	=	B

7. ### Run the test

   Run the test and check the results.

How to Use Flow Control Activities

Relevant for: API testing only

This task describes how to use Flow Control activities in an API test.

This task includes the following steps:

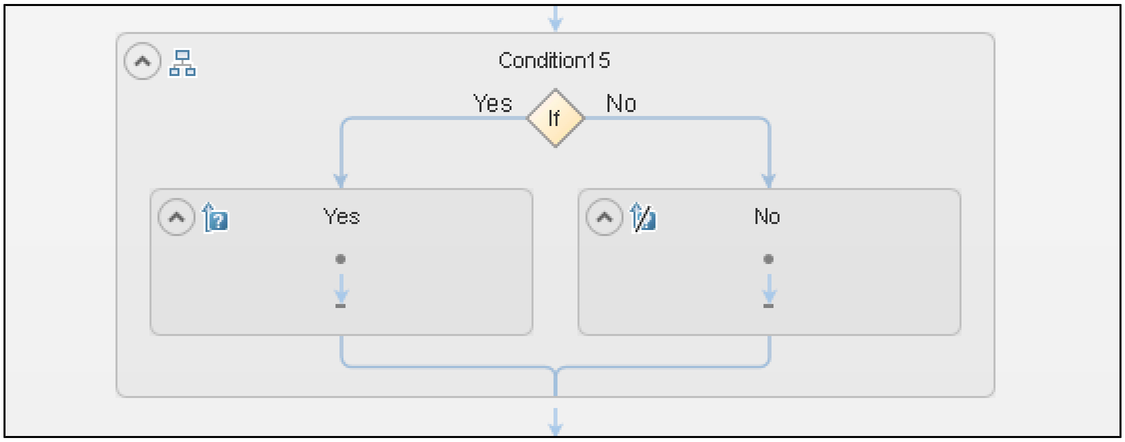
- "Add a conditional step" below
- "Add a loop" on the next page
- "Add steps to pause and restart the test" on page 1622
- "Add a wait step" on page 1622

Add a conditional step

Conditional steps enable you to specify alternate test paths depending on the outcome of a previous step. This is the equivalent of an If...Else function in an application's code.

1. In the Toolbox pane, expand the **Flow Control** activities node and drag a **Condition** step to the canvas.

   The canvas displays a branched test step, with **Yes** and **No** branches, as see in the example below:

2. Set the trigger for each branch of the step.

   You can the trigger for the **Yes** or **No** branches in the following ways:

   - Specify a condition for the output of a test step.

     i. In the **Condition** tab in the Properties pane, select the **Use condition** option. The condition properties are displayed.

     ii. In the **Variable** field, click the **Link to data source** button .

     iii. In the Select Link Source dialog box, link to the output of a previous step. For details on linking properties to other steps, see "Link your test step to another step" on page 1820.

     > **Note:** You must link the **Variable** field to a previous step to set the value to meet

to trigger the condition.

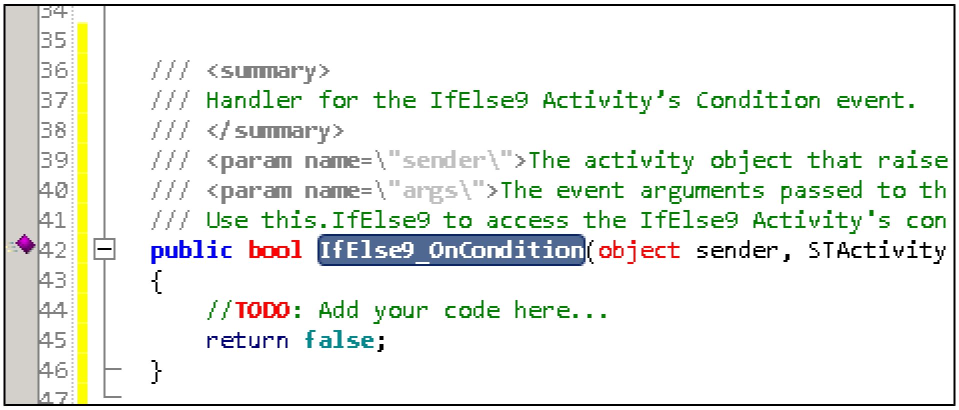
iv. Specify the expected **Value** and the **Operator** (equal to, greater than or less than, etc.) for the step selected in the **Variable** field.

- Use an event handler.

  i. In the **Condition** tab in the Properties pane, select the **Use event** option.

  ii. In the **Events** tab , in the **Condition** row, click the drop down arrow and select **Create a default handler**. The TestUserCode.cs file opens in the document pane.

  iii. In the **IfElse9_OnCondition** section of the TestUserCode.cs file, replace the **TODO** section with your event handler.



```
    /// <summary>
    /// Handler for the IfElse9 Activity’s Condition event.
    /// </summary>
    /// <param name=\"sender\">The activity object that raise
    /// <param name=\"args\">The event arguments passed to th
    /// Use this.IfElse9 to access the IfElse9 Activity's con
    public bool IfElse9_OnCondition(object sender, STActivity
    {
        //TODO: Add your code here...
        return false;
    }
```

3. Drag activities to the **Yes** and **No** branches of the Condition step to create the steps to run based on the results of the condition.

Add a loop

Loops enable you to run the steps contained in a the loop multiple times.

1. In the Toolbox pane, expand the **Flow Control** activities node and drag a **Loop** step to the canvas.

2. In the **Input** tab of the Properties pane, select the Loop type and define the loop properties.

   Select one of the following loop types:

   - **'For' Loop**

     This loop runs the included steps the specified number of times. Set the number of iterations to run the loop.

> **Note:** If the number of iterations is defined elsewhere (the result of a previous step or a data table), link to the property by clicking the **Link to data source** button in the **Value** cell for the **Number of Iterations** property.

- **'Do While' Loop**

  This loop runs indefinitely until a specific condition is met. When the condition is met, the test advances to the steps following the loop.

  You can set the loop run properties in the following ways:

  - Specify a condition.

    - In the **Condition** tab in the Properties pane, select the **Use condition** option. The condition properties are displayed

    - In the **Variable** field, click the **Link to data source** button .

    - In the Select Link Source dialog box, link to the output of a previous step.

      > **Note:** You must link the **Variable** field to a previous step to set the value to meet to trigger the condition.

    - Specify the **Value** and the **Operator** (equal to, greater than or less than, etc.) for the value of the step selected in the Variable field.

  - Use an event handler.

    - In the **Condition** tab in the Properties pane, select the **Use event** option.

    - In the **Events** tab , in the Condition row, click the drop down arrow and select **Create a default handler**. The TestUserCode.cs file opens in the document pane.

    - In the **Loop_OnCondition** section of the TestUserCode.cs file, replace the **TODO** section with your event handler.

- **'For Each' Loop**

  This loop runs one time for each item in a selected collection (usually an array). To link this loop to a collection from a different step, click the **Link to data source** button and select the collection in the Select Link Source dialog box.

3. Drag activities inside the loop.

4. Associate data sources with the loop.

You can assign a specific data source to use with the current loop. The data from this data source is then available for all steps included in the loop. For details, see "Add a data source to the Test Flow or test loop" on page 1826.

Add steps to pause and restart the test

You can add any of the following steps to your test to pause and restart the test:

- **Break.** This step stops the test. You insert a **Continue** step to resume the test.
- **Continue.** This step resumes the test after a **Break** step stops it.
- **Sleep.** This step temporarily pauses the test for a specified number of milliseconds. Enter the number of milliseconds to wait in the **Input/Checkpoints** tab for the step.

Add a wait step

Wait steps are mandatory when your application uses an asynchronous service call. After the call to the service, you insert the Wait step. While the test is waiting for the response from the service call, the test waits the amount of time specified in this step.

1. In the Toolbox pane, expand the **Flow Control** activities node and drag a **Wait** step to the canvas.
2. In the **Input/Checkpoints** tab in the Properties pane, Set the step properties:
   - **Timeout:** the number of milliseconds to pause the test.
   - **Start timeout from step:** the step for which UFT is waiting for a response.
   - **Action on Timeout:** what happens when the end of the timeout is reached and the step's response is not received.
   - **Completion events:** the events that signify completion of the timeout.

How to Use Date and Time Activities

Relevant for: API testing only

The following describes various tasks that you can perform with the Date/Time activities, such as incrementing a date or finding a differential.

This task includes:

- "Increment a date and time" on the next page
- "Find the difference between two date/time expressions" on the next page
- "Format a date or time" on page 1624

Increment a date and time

1. Drag the **Increment Date/Time** activity onto the canvas.

2. Click the Input/Checkpoints tab in the Properties pane.

3. Set the Input property—**Original Date/Time**. Click the arrow in the **Value** column to open the calendar. Click **Today** or select another date. You can also link to the output of another step using the **Link to a data source** button.

4. To set a time unit such as hour, minute, seconds, or milliseconds, edit the expression in the **Original Date/Time** row. For milliseconds, add a colon followed by a three digit millisecond value, for example, 2011-01-01T01:30:00:040.

5. Set the Input property—**Unit**. Select a unit by which to increment: Milliseconds, Seconds, Minutes, Hours, Days, Months, or Years.

6. Set the Input property—**Amount**. Use the scroller to set the amount of units to increment.

7. Select the **Result** check box in the **Checkpoints** pane to verify the response. Use the calendar to select the expected date. If you need to set an expected value for a time unit, modify the time units manually.

Find the difference between two date/time expressions

1. Drag the **Date/Time Difference** activity onto the canvas.

2. Click the Input/Checkpoints tab in the Properties pane.

3. Set the Input property— **Date/Time A**. Click the arrow in the **Value** column to open the calendar. Click **Today** or select another date.

4. To set a time unit such as hour, minute, seconds, or milliseconds, edit the expression in the **Date/Time A** row. For milliseconds, add a colon followed by a three digit millisecond value, for example, 2011-01-01T01:30:00:040.

5. Set the Input property—**Date/Time B**. Click the arrow in the **Value** column to open the calendar. Select a date for the second expression.

   To set a time unit such as hour, minute, seconds, or milliseconds, follow the instructions in the above step.

6. Set the Input property—**Unit**. Select the unit by which you want to measure the difference: Milliseconds, Seconds, Minutes, Hours, Days, Months, or Years.

7. To verify the response, select the **Difference** check box in the **Checkpoints** pane. Use the scroller to specify the expected difference.

Format a date or time

1. Drag the **Format Date/Time** activity onto the canvas.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Set the property **Input Date/Time**, the term to which you want to apply formatting. Click the arrow in the **Value** column to open the calendar. Click the date displayed at the top of the calendar window to use the current date, or select another date.
4. If you provide a date/time in a string (non-date) format, select a format in the **Input Date/Time Format** row. Otherwise, leave this row empty.
5. Set the **Format** input property. Select a format from the drop down or type in a custom format for the output expression. If your expression contains milliseconds, use an expression with a time format and add a colon followed by fff, for example, dd/MM/yyyy hh:mm:ss:fff.
6. Select the **Result** check box in the **Checkpoints** pane to verify the response's format. Type the expected expression. If you need to set an expected value for a time unit, modify the time units manually.

How to Execute Database Commands or Retrieve Data

Relevant for: API testing only

The following describes how to use database activities. For details about the database activities, see "Database Activities" on page 1669.

This task includes:

- "Open a connection" below
- "Add a Select Data step" on the next page
- "Add an Execute Command step" on the next page
- "Add database transaction activities" on the next page
- "Add a Close Connection step" on page 1626

Open a connection

This step is mandatory for all of the following steps.

1. Drag the **Database > Open Connection** activity onto the canvas.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Select the Input property—**Connection string**.
4. Click the **Connection Builder** button in the right side of the **Value** column ... . The

Connection Builder dialog box opens. For details, see "Connection Builder Dialog Box" on page 1720.

5. Paste in an existing string into the text area or click the **Connection Builder** button to open Microsoft's Data Link Properties dialog box. Provide the required information and click **OK**. For details, click the dialog box's **Help** button.

   > **Note:** The database connection must be an OLE DB type.

6. If you want to validate the connection, set the checkpoint properties - **Results** and **Results message**.

Add a Select Data step

1. Drag the **Database > Select Data** activity onto the canvas, below an **Open connection** step.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Set the Input property—**Connection**. Click the **Link to a data source** button by the right side of the **Value** column . In the Select Link Source dialog box, select an earlier **Open connection** step in the left pane. In the right pane, select the step's result.
4. Set the Input property—**Query string**. Click the **Browse** button in the right side of the **Value** column to open the Query Builder. Paste in a query or use the Query Designer. For details, see the "Query Builder Dialog Box" on page 1722.
5. Optionally, set the **Timeout**, the maximum time allowed for the database response. To allow an unlimited amount of time, enter 0.

Add an Execute Command step

1. Drag the **Database > Execute Command** activity onto the canvas, below the **Open Connection** step.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Set the Input property—**Connection**. Click the **Link to a data source** button by the right side of the **Value** column . In the Select Link Source dialog box, select an earlier **Open connection** step in the left pane. In the right pane, select the step's result.
4. Set the Input property—**Command**. Click the arrow to open an edit box, Paste in a command and click **OK**.
5. Optionally, set the **Timeout**, the maximum time allowed for the database response.

Add database transaction activities

You can optionally add transaction activities to your test.

1. Drag the **Database > Begin Transaction** activity to the canvas after an **Open Connection** step and before the steps to be included in the transaction.

2. Drag a **Database > Commit Transaction** activity to the canvas after the database steps that make up the transaction.

3. Drag a **Database > Rollback Transaction** activity to the canvas after the database steps that make up the transaction.

> **Tip:** A common use is to insert a **Condition** step to check a value after the database commands. Set one branch with **Commit Transaction** and the other branch to **Rollback Transaction**. For details, see "Flow Control (API Testing only)" on page 210.

Add a Close Connection step

1. Drag the **Database > Close Connection** activity on to the canvas, after all of the database steps.

2. Click the Input/Checkpoints tab in the Properties pane.

3. Set the Input property—**Connection**. Click the **Link to a data source** button in the right side of the **Value** column 🔗 . In the Select Link Source dialog box, select the **Available steps** option. In the left pane, select the earlier step, **Open Connection**, In the right pane, select the **Connection** node.

How to Send a Multipart HTTP Request

Relevant for: API testing only

The following describes how to send an HTTP request that uses multiple parts. For details about the properties, see the "Multipart Tab (Properties Pane - API Testing)" on page 426.

This task includes:

- "Add an HTTP Step" below
- "Set the General properties" below
- "Set the properties for the first part" on the next page
- "Set the properties for the next parts" on the next page

1. Add an HTTP Step

Drag the **Network > HTTP** activity onto the canvas.

2. Set the General properties

Open the General tab

in the Properties pane. Set the properties as described in the "General Tab (Properties Pane - API Testing)" on page 413.

3. **Set the properties for the first part**

   Open the **Input/Checkpoints** tab in the Properties pane. Set the properties for the first part as described in the "Network Activities" on page 1684.

4. **Set the properties for the next parts**

   a. Open the **Multipart** tab in the Properties pane.

   b. Select the **Enable Multipart** option.

   c. Set the multipart type and global header information.

   d. Add elements to the Parts array corresponding to the number of parts.

   e. Set the properties for the parts as described in the "Multipart Tab (Properties Pane - API Testing)" on page 426.

   f. If you want to validate a response, select the box in the **Validate** column and provide the expected values.

How to Create a Call to a Java Class

Relevant for: API testing only

The Call Java Class activity lets you to add Java steps to your test script. This feature enables you to incorporate existing Java code into your test.

This task describes how to create the Java call and includes the following steps:

- "Set the global Java settings - optional" on the next page
- "Implement the Java interface" on the next page
- "Compile the Java source code" on the next page
- "Package your custom step - optional" on the next page
- "Set up the Java environment - optional" on the next page
- "Add a Call Java Class activity" on page 1629
- "Open the Java Class Setting dialog box" on page 1629
- "Set the Java Call properties" on page 1629
- "Provide Input property values" on page 1629

1. Set the global Java settings - optional

To set global VM (Virtual Machine) settings, select the Start or End steps in the canvas, and open the **Test Settings** view in the Properties pane. For details, see the "Test Settings Tab (Properties Pane - API Testing)" on page 429.

2. Implement the Java interface

Change to the <installation_folder>\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java folder and create an implementation for the java interface. For an example, see the sample subfolder.

This interface includes the essential information for the Java call, such as input properties, output properties, and a point of entry. The following methods are included:

- **getInputProperties.** Returns a mapping of the input property names and their Java class.
- **getOutputProperties.** Returns a mapping of the output property names and their Java class.
- **Execute.** A method that receives the mapping of the input property names and their actual values i.e. their object instance. In this method, you process input properties and delegate them to your own Java artifacts. Afterward you process the output properties and send their mappings and their actual values as the method's output.

3. Compile the Java source code

Compile the java files located in the <installation_folder>\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java folder.

> **Tip:** To determine which JDK to use for, check the version of Java JRE installed with UFT. Open the <installation_folder>/jre/bin folder and right-click the java.exe file. Select **Properties** and open the **Version** tab.

4. Package your custom step - optional

Package your java classes into a .jar file. This is optional, since you can also provide a package root for the class.

5. Set up the Java environment - optional

a. Click in a Start or End step and click the **Test Settings** tab in the Properties pane. Set the Java test settings for the VM and JMS. For details, see the "Test Settings Tab (Properties Pane - API Testing)" on page 429.

b. Expand the **JMS** node in the Toolbox pane and drag a **JMS** activity onto the canvas.

c. Set the step's properties. Click on the step in the canvas, and select the Input/Checkpoints tab in the Properties pane. Enter a value for Queue, Subscription, Topic name, and any other relevant property.

d. For Send activities, specify a message.

e. For Receive activities, select the output properties you want to validate in the **Checkpoints** pane and specify their values.

For more details, see "JMS Activities" on page 1691.

6. Add a Call Java Class activity

Expand the **Java** node in the Toolbox pane, and drag the **Call Java Class** activity onto the canvas.

7. Open the Java Class Setting dialog box

Select the Java step in the canvas, and click the Input/Checkpoints tab in the Properties pane. Click the **Java Class** button to open the "Java Class Dialog Box" (described on page 1727.

If you need to embed the jar in the script, you must select the check box before browsing for and selecting the class file.

8. Set the Java Call properties

In the Java Class dialog box:

a. Provide a classpath. If you packaged your Java step, click the **Browse** button adjacent to the **Jar** field and point to a .jar file. Alternatively, click the **Browse** button adjacent to the **Package root** field and point to a package root folder. To embed the jar and save it with the test, select **Embed Jar in Test**. Due to a technological limitation, if you intend to specify a class file, you must select the **Embed Jar in Test** option before you browse for the class file.

b. Click the **Browse** button adjacent to the **Class file** field to locate the class within the .jar file or the folder. Make sure it is a class that implements the **ServiceTestCall** interface.

c. To provide additional classpaths, click the **Jar** or **Folder** buttons in the **Additional Classpaths** section and browse to a .jar file or classpath folder. Click **Add** to move the contents into the list.

d. Click **OK** to save the Java Call settings.

For user interface details, see the "Java Class Dialog Box" on page 1727.

9. Provide Input property values

In the Properties pane's Input/Checkpoints tab , provide values for the step's input properties.

How to Retrieve Messages from a JMS Queue

Relevant for: API testing only

This task describes how to send, receive and browse JMS messages on a JMS queue.

This task includes the following steps:

- "Define the global JMS settings" below
- "Prepare a message to send to the queue" below
- "Add a Send Message step to your test" below
- "Add a Send Message with security and attachments - optional" on the next page
- "Add a Receive Message step to your test - optional" on the next page
- "Browse the message queue - optional" on the next page
- "Set checkpoints- optional" on page 1632
- "Run the test and view the run results" on page 1633

1. Define the global JMS settings

Define global JMS settings such as the JNDI details according to the specifications of your JMS messaging server. Select the **Start** or **End** step in the canvas and open the **Test Settings** view in the Properties pane. For details, see the "Test Settings Tab (Properties Pane - API Testing)" on page 429.

2. Prepare a message to send to the queue

Prepare the message you want to send to the queue in one of the following ways:

- Prepare an expression and type it into the **Message** property area.
- Use other activities to generate the string, for example **Concatenate String**, **Trim String**, and so forth.

3. Add a Send Message step to your test

a. In the Toolbox pane, expand the **Standard Activities > JMS** node. Drag a **Send Message to JMS Queue** or a **Send and Receive Message from JMS Queue** activity onto the canvas.

b. Select the step and click the Properties pane's **Input/Checkpoints** tab. Set the send-related input properties:

- **Send queue name**
- **Message**
- **JMS send properties**

c. If you used another step to generate the message text, link its output to the **Message** property.

4. **Add a Send Message with security and attachments - optional**

   To send a Web service message via JMS with attachments and security setting, such as tokens and signatures:

   a. Drag in or select the Web service step in the canvas and disable its **Send Request to Service** option in the Properties pane. For details, see "HTTP Request Activity" on page 1684 and "SOAP Request Activity" on page 1688.

   b. Set the security options for the service. For details, see "How to Set Security for a Web Service on the Port Level" on page 1885.

   c. Include any attachments. For details, see "Attachments Tab (Properties Pane - API Testing)" on page 403.

   d. Drag in a Send Message to JMS Queue activity onto the canvas.

   e. In the Properties pane's Input/Checkpoints tab, select the link icon 🔗 for the **Message** property. The Select Link Source dialog box opens. For details, see the "Select Link Source Dialog Box (API Testing)" on page 1840.

   f. In the Select Link Source dialog box, select the Web service step in the left pane. In the Output property section, double-click on the **Raw Request** property. This attaches the security and attachment data to the message.

   g. Define any other JMS properties as you normally would. For property details, see "JMS Activities" on page 1691.

5. **Add a Receive Message step to your test - optional**

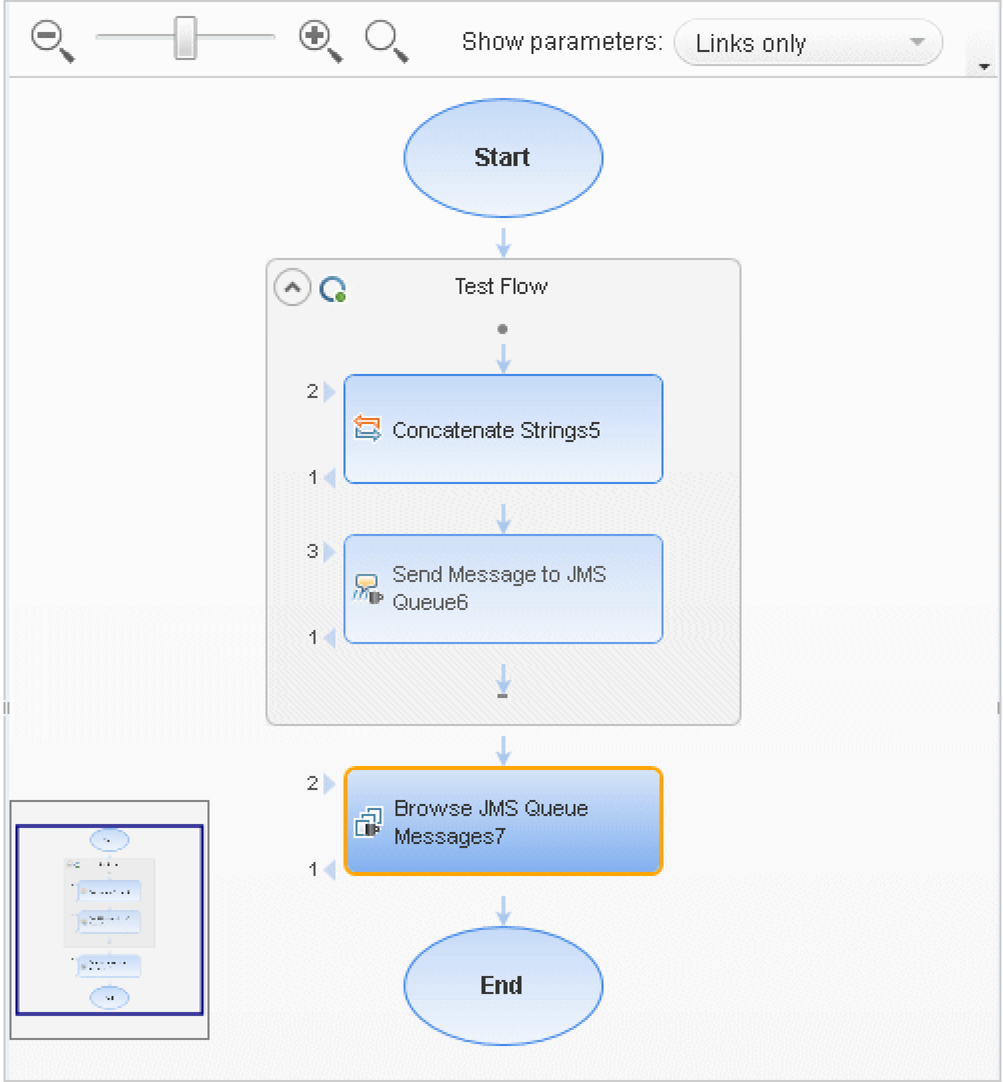
   a. To retrieve a message from the queue, drag a **Receive Message from JMS Queue** activity onto the canvas. If you added a **Send and Receive Message to JMS Queue** activity in the previous step, you can skip this step.

   b. Select the **Receive Message from JMS Queue** step and open the Properties pane's Input/Checkpoints tab. Set the receive-related input properties: **Receive queue name** and **JMS receive message selector**. For property details, see "JMS Activities" on page 1691.

6. **Browse the message queue - optional**

   a. To browse the messages in the queue without consuming them, drag a **Browse JMS**

Queue Messages activity onto the canvas, outside of the Test Flow. Make sure that a **Send (and Receive) Message from JMS Queue** step precedes the **Browse JMS Queue Messages** step.



b. Click the Properties pane's Input/Checkpoints tab, and select the **Browse JMS Queue Messages** step. Set the browse-related input properties:

- **Queue name.** Provide a queue name or link to the queue name from an earlier step. For IBM Websphere's MQ, if you specify a queue name that does not exist, a new queue will be created during the test run.

- **JMS receive message selector.** The selector lets you filter the messages list. For details about these properties, see "JMS Activities" on page 1691.

7. Set checkpoints- optional

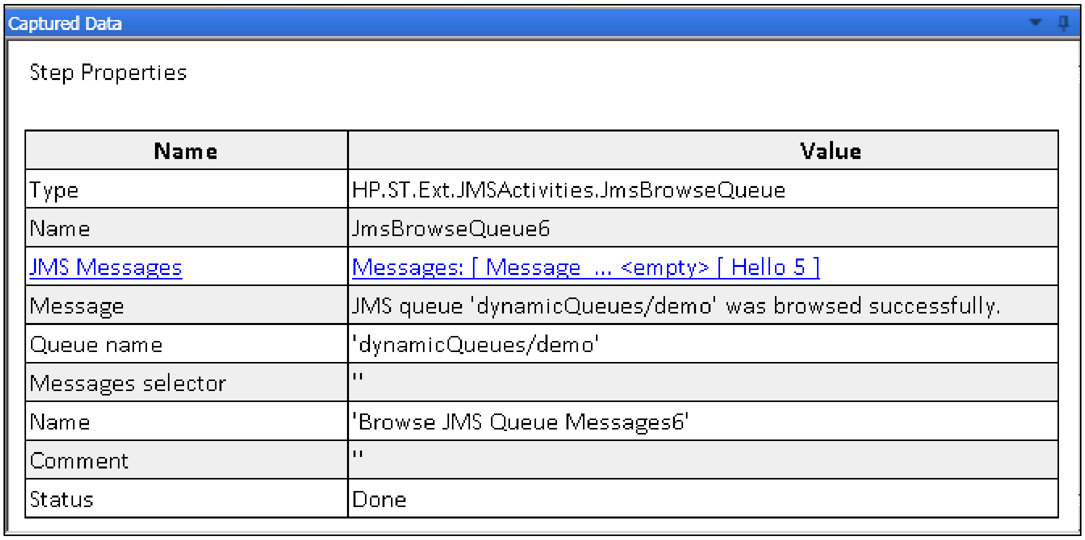
a. Select the **Browse JMS Queue Messages** step and click in the Properties pane's **Checkpoints** section.

b. Select the properties you want to validate and provide values.

c. Use the **Browse JMS queue messages** output properties to validate the messages on the queue. For example, you can check the value of a specific property or check the number of messages on the queue.

8. Run the test and view the run results

Run the test. In the Run Results Viewer, expand the checkpoint nodes and verify that the actual values match the expected values. Click the link in the report to display the JMS queue messages in a separate browser. Compare these messages with those shown in the queue on the JMS application server's console.



Step Properties

Name	Value
Type	HP.ST.Ext.JMSActivities.JmsBrowseQueue
Name	JmsBrowseQueue6
JMS Messages	Messages: [Message ... <empty> [Hello 5]
Message	JMS queue 'dynamicQueues/demo' was browsed successfully.
Queue name	'dynamicQueues/demo'
Messages selector	''
Name	'Browse JMS Queue Messages6'
Comment	''
Status	Done

How to Receive Messages Through JMS Topics

Relevant for: API testing only

This task describes how to work with subscriptions to JMS topics.

This task includes the following steps:

- "Set the global JMS settings" on the next page
- "Subscribe to a topic" on the next page
- "Publish a message to the topic" on the next page
- "Include Web service security and attachments - optional" on the next page
- "Receive the message through the topic" on the next page

1. **Set the global JMS settings**

   Set global JMS settings such as the JNDI details according to the specifications of your JMS messaging server. Click in the canvas outside of the Test Flow, and open the **Test Settings** tab in the Properties pane. For details, see the "Test Settings Tab (Properties Pane - API Testing)" on page 429.

2. **Subscribe to a topic**

   Add a **Subscribe to JMS Topic** step and provide values for its input properties as described in "JMS Activities" on page 1691.

3. **Publish a message to the topic**

   Add a **Publish Message to JMS Topic** step and provide values for its input properties.

4. **Include Web service security and attachments - optional**

   To publish a Web service message via JMS with attachments and security setting, such as tokens and signatures:

   a. Drag in or select the Web service step in the canvas and disable its **Send Request to Service** option in the Properties pane. For details, see "HTTP Request Activity" on page 1684 and "SOAP Request Activity" on page 1688.

   b. Set the security options for the service. For details, see "How to Set Security for a Web Service on the Port Level" on page 1885.

   c. Include any attachments. For details, see "Attachments Tab (Properties Pane - API Testing)" on page 403.

   d. Drag a **Publish Message to JMS Topic** step onto the canvas.

   e. In the Properties pane's Input/Checkpoints tab, select the link icon for the **Message** property. The Select Link Source dialog box opens. For details, see the "Select Link Source Dialog Box (API Testing)" on page 1840.

   f. In the Select Link Source dialog box, select the Web service step in the left pane. In the Output property section, double-click on the **Raw Request** property. This attaches the security and attachment data to the message.

   g. Define any other JMS properties as you normally would. For property details, see "JMS Activities" on page 1691.

5. **Receive the message through the topic**

   Drag a **Receive Message from JMS Topic** activity onto the canvas. Use the topic and subscription name that you defined in the previous steps.

How to Retrieve Messages from an MQ Queue

Relevant for: API testing only

This task describes how to work with Put and Get on an IBM Websphere MQ queue.

This task includes the following steps:

- "Prerequisite" below
- "Connect to the MQ Queue Manager" below
- "Put a message on the queue" below
- "Browse the message queue - optional" below
- "Get a message from the queue" on the next page
- "Commit or Backout the actions - optional" on the next page
- "Disconnect from the MQ Queue Manager" on the next page
- "Define event handlers - optional" on the next page
- "Set checkpoints - optional" on the next page

1. Prerequisite

You must have the MQ client installed on all machines upon which you want to run the test.

2. Connect to the MQ Queue Manager

A connection step must precede all steps that use the specified Queue Manager. Drag an **IBM Websphere MQ > Connect to MQ Queue Manager** step onto the canvas and provide the connection details in the Properties pane's Input/Checkpoints tab. For details see "Put Message to MQ Queue Activity" on page 1704.

3. Put a message on the queue

Drag a **Put Message to MQ Queue** step onto the canvas, below the connection step. The canvas automatically links the **MQManager** property to the most recent **Connect to MQ Queue Manager** step. Set input values as described in "Put Message to MQ Queue Activity" on page 1704.

4. Browse the message queue - optional

To browse the messages in the queue without consuming them:

a. Drag a **Browse Messages in MQ Queue** activity onto the canvas.

b. Select the step in the canvas and open the Properties pane's **Input/Checkpoints** tab.

c. Set the browse-related properties. For details, see "Browse Messages in MQ Queue Activity" on page 1702.

5. **Get a message from the queue**

   Drag a **Get Message from MQ Queue** step onto the canvas, below the connection step. Set the input values as described in "Get Message from MQ Queue Activity" on page 1705.

6. **Commit or Backout the actions - optional**

   To commit the actions performed until a certain point, drag a **Commit MQ Pending Messages** activity onto the canvas. To roll back changes drag a **Backout MQ Pending Messages** activity onto the canvas at the relevant location.

7. **Disconnect from the MQ Queue Manager**

   A disconnection step must follow all steps that use the specified Queue Manager. Drag a **Disconnect from MQ Queue Manager** step onto the canvas, and provide the connection details in the Properties pane's **Input/Checkpoints** tab.

8. **Define event handlers - optional**

   To customize or automate the actions against the MQ server, use the MQ event handlers. Click the Events button in the Properties pane and double-click the appropriate event—a standard event or one of the MQ-specific events, such as BeforeMQGetMessage or AfterMQGetMessage. Customize the code in the TestUserCode.cs tab. For details, see "Writing Event Handlers for API Test Steps" on page 1935 and the "Events Tab (Properties Pane - API Testing)" on page 409.

9. **Set checkpoints - optional**

   To validate the response data, select the step and click within the Properties pane's **Checkpoints** section. Indicate the properties you want to validate and provide values. Run the test and view the run results.

How to Receive Messages Through MQ Topics

Relevant for: API testing only

This task describes how to retrieve messages published to MQ topics.

This task includes the following steps:

- "Prerequisite" on the next page
- "Connect to the MQ Queue Manager" on the next page
- "Subscribe to a topic" on the next page
- "Publish a message to the topic" on the next page

- "Receive the message through the topic" below
- "Unsubscribe from a topic" below
- "Disconnect from the MQ Queue manager" below
- "Define event handlers - optional" below

1. **Prerequisite**

   You must have the MQ client installed on all machines upon which you want to run the test.

2. **Connect to the MQ Queue Manager**

   A connection step must precede all steps that use the specified Queue Manager. Drag a **Connect to MQ Queue Manager** step onto the canvas, and provide the connection details in the Properties pane's **Input/Checkpoints** tab. For details see "Connect to MQ Queue Manager Activity" on page 1700.

3. **Subscribe to a topic**

   Add a **Subscribe to MQ Topic** step. The canvas automatically links the **MQManager** property to the most recent **Connect to MQ Queue Manager** step. Provide values for its input properties as described in "Subscribe to MQ Topic Activity" on page 1709.

4. **Publish a message to the topic**

   Add a **Publish Message to MQ Topic** step and provide values for its input properties as described in "Publish Message to MQ Topic Activity" on page 1711.

5. **Receive the message through the topic**

   Drag a **Receive Message from MQ Topic** activity onto the canvas. Use the topic and subscription name that you set in the previous steps.

6. **Unsubscribe from a topic**

   Add an **Unsubscribe from MQ Topic** step and provide values for its input properties as described in "Unsubscribe from MQ Topic Activity" on page 1710.

7. **Disconnect from the MQ Queue manager**

   A disconnection step must follow all steps that use the specified Queue Manager. Drag a **Disconnect from MQ Queue Manager** activity onto the canvas. The canvas automatically links to the most recent opened connection.

8. **Define event handlers - optional**

   To customize or automate the actions against the MQ server, use the MQ event handlers. Click the **Events** button in the Properties pane and double-click the appropriate event—a standard event or one of the MQ-specific ones. Customize the code in the TestUserCode.cs tab. For details, see "Writing Event Handlers for API Test Steps" on page 1935 and the "Events Tab (Properties Pane - API Testing)" on page 409.

How to Create an SAP API Test Step

Relevant for: API testing only

This task describes how to create a test step for an SAP IDoc or RFC.

This task includes the following steps:

- "Prerequisite" below
- "Define an SAP Connection" below
- "Open the Import from SAP dialog box" below
- "Select a connection" below
- "Provide credentials - optional" below
- "Search for an IDoc or RFC" on the next page
- "Add the IDoc or RFC to the Toolbox pane" on the next page
- "Create a test/action step" on the next page

1. **Prerequisite**

   You must have the SAP .NET Connector installed on your machine. The installation is available on the SAP Help portal, at http://help.sap.com/saphelp_NW04/helpdata/en/e9/23c80d66d08c4c8c044a3ea11ca90f/content.htm.

2. **Define an SAP Connection**

   Select **Tools > Options > API Testing** tab > **SAP Connections** node. In the SAP Connections pane, define one or more SAP connections. For details, see "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 568.

3. **Open the Import from SAP dialog box**

   Select **Tools > Import from SAP**. For user interface details, see "Import from SAP/Update Dialog Box" on page 1729.

4. **Select a connection**

   Select one of the connections that you defined in the SAP Connections pane of the Options dialog box.

5. **Provide credentials - optional**

   Accept the default credentials (as entered in the SAP Connections pane in the Options dialog box) or click **Override connection** to provide different information.

6. **Search for an IDoc or RFC**

   Select **IDoc** or **RFC** and specify a search string using an asterisk (*) as a wildcard. Click **Search**.

7. **Add the IDoc or RFC to the Toolbox pane**

   Check the returned items and click **Import Selected** to add them to the **SAP** category in the Toolbox pane.

8. **Create a test/action step**

   Expand the **Local Activities > SAP > <Connection Name> > IDOCs/ RFCs** node in the Toolbox pane, and drag an SAP activity to the canvas. Provide values for the **General** and **Input/Checkpoints** properties.

How to Call External Tests or Actions

Relevant for: API testing only

This task describes how to incorporate tests from other HP applications. You can call any of the following types of tests:

- Unified Functional Testing tests (both GUI and API tests)
- QuickTest Professional tests
- Service Test tests
- VuGen (Virtual User Generator) scripts from HP LoadRunner

For details on integration between UFT API Testing and other HP products, see "Automated Testing Tool Integration" on page 1596.

This task includes the following steps:

- "Prerequisites" on the next page
- "Call an API Test or Action or Service Test test" on the next page
- "Call a GUI Test or QuickTest action or test" on page 1641
- "Add a LoadRunner script activity" on page 1642

1. **Prerequisites**

   Make sure you have installed the application whose test/script you want to call on the same computer with UFT or have access to the directory containing the tests or scripts.

   > **Notes:**
   >
   > - If you are calling a test last modified in a version of Service Test prior to your present version, make sure the test was modified with Service Test 11.10 or higher or UFT.
   > - To call a test created in QuickTest Professional, ensure that the test was created with QuickTest 11.00. The first time you run the step, you may need to wait several seconds for UFT to invoke and load the test.
   > - If you are calling a test created in HP LoadRunner, ensure that you created or opened the test in LoadRunner version 11.00 or later.

2. **Call an API Test or Action or Service Test test**

   a. Make sure the action or test you want to call has been saved and run successfully at least once.

   b. In the **Standard Activities** section of the Toolbox pane, expand the **HP Automated Testing Tools** node.

   c. Drag the **Call API Action or Test** activity onto the canvas.

   d. In the **Input/Output Properties** tab in the Properties pane, click the **Select Action or Test** button.

   e. In the "Select Action or Test Dialog Box", select a test last modified with Service Test 11.10 or higher, or with UFT.

   > **Note:** To refresh this test or action call when you are editing the test, click the **Refresh** button in the Input/Output Properties tab in the Properties pane..

   f. In the Input/Output Properties tab, edit the property values as needed.

   > **Note:**
   >
   > - The property list remains empty until you select a test.
   > - If the test you are calling has no input or output parameters, the Input/Output Properties tab will be empty.

   g. Add other relevant steps to your test. You can link input properties of subsequent step to the output properties of the step containing the called API test or action..

h. If the value of the input parameter for step containing the API test/action call must be a string (such when the result of a previous step was XML), add an **XML to String** activity before the step containing the call to the action or test.

i. Optional - To specify a custom directory for the results, in the General tab of the Properties, click the **Browse** button ... in the **Results directory** row in the **General** view tab.

j. Save and run the test.

3. Call a GUI Test or QuickTest action or test

a. Make sure the action or test you want to call has been saved and run successfully at least once.

b. In the **Standard Activities** section of the Toolbox pane, expand the **HP Automated Testing Tools** node.

c. Drag the **Call GUI Action or Test** activity onto the canvas.

> **Note:** This activity is only available when working with an HP Unified Functional Testing license.

> **Tip:** Do not insert a call to a QuickTest or GUI action or test that contains a call to an API Test action or test, as this can cause unexpected behavior.

d. In the **Input/Output Properties** tab in the Properties pane, click the **Select Action or Test** button. In the "Select Action or Test Dialog Box" (described on page 1797, select an action or a test.

e. In the "Select Action or Test Dialog Box", select an action or a test created in QuickTest 11.00 or UFT.

> **Note:** If you want to use data from a GUI test in your API test, the GUI test or action that is called must have test or action parameters saved with the test or action.

> **Tip:** If the QuickTest or GUI action or test changed at its source, you can reload it. In the Input/Output Properties tab, click **Refresh**.

f. In the Input/Output Properties tab, edit the property values as needed. If you want to use parameters from the called GUI test, in the Input/Checkpoints tab, click the **Link to data source** button in subsequent test steps. In the Select Link Source dialog box, link the selected property to a GUI test or action parameter.

> **Note:**
>
> - The property list remains empty until you select a test.
>
> - If the test you are calling has no input or output parameters, the Input/Output Properties tab will be empty.

g. Add other relevant steps to your test.

h. Save and run the test.

4. **Add a LoadRunner script activity**

a. Make sure the action or test you want to call has been saved and run successfully at least once.

b. In the **Standard Activities** section of the Toolbox pane, expand the **HP Automated Testing Tools** node.

c. Drag the **Call Virtual User Generator Script** activity onto the canvas.

d. Open the General tab in the Properties pane, and click the script selection button .

e. Navigate to the directory where your VuGen script file (.usr) is saved.

f. Add other relevant steps to your test.

g. Save and run the test.

How to Prepare and Run a Load Test

Relevant for: API testing only

This task describes how to prepare a test for load testing in LoadRunner. For an explanation about load testing, see "Load Testing Activities" on page 1698.

This task includes the following steps:

- "Prerequisite" on the next page
- "Create a load test" on the next page
- "Add test steps" on the next page

- "Prepare for load testing - optional" below
- "Assign data to the test - optional" on the next page
- "Set the data retrieval properties - optional" on the next page
- "Set the run configuration to Release" on the next page
- "Run in Load Testing mode to validate the test" on the next page
- "Incorporate the test into LoadRunner" on the next page

1. Prerequisite

Make sure that HP LoadRunner or a standalone version of VuGen (HP Virtual User Generator) is installed. Without this installation, the Load Testing template will not be available.

2. Create a load test

In the "New <Document> Dialog Box" on page 152 (described on page 152), in the **Select type** section, choose **API Load Test**.

If you have a test that was created with the standard API testing template, select **Design > Operation > Enable Test for Load Testing** or click the **Enable Test for Load Testing** button .

3. Add test steps

Drag activities from the Toolbox pane onto the canvas to add steps to the test.

4. Prepare for load testing - optional

To measure the performance of a group of steps, define a transaction.

a. Mark the beginning of a transaction.

   - Drag the **Start Transaction** activity from the Toolbox pane's **Load Testing** category, onto the canvas. Place it before the first step of the group of steps that you want to measure.
   - Click the Input/Checkpoints tab in the Properties pane. Enter a **Transaction name**. This name will be used in LoadRunner Analysis.

b. Mark the end of a transaction.

   - Drag the **End Transaction** activity to the end of the group of steps you want to measure.
   - In the Properties pane's Input/Checkpoints tab, type a transaction name. The name must be one that was already used for a prior **Start Transaction** step.

- In the End Transaction's **Input** properties, select a **Status** for reporting: PASS, FAIL, AUTO, or STOP.

  > **Note:** The End Transaction status is only the LoadRunner transaction's status—not the status of step in UFT. For example, if you assign a **Failed** status to the transaction, UFT can still issue a **Passed** status for the test step.

c. Set the think time.

- If you want to emulate think time, drag the **Load Testing >Think Time** activity between the relevant steps.
- In the Properties pane's Input/Checkpoints tab, click in the **Duration (sec)** row and specify a think time in seconds.

5. Assign data to the test - optional

Import or create data tables for the input properties. For details, see "How to Assign Data to API Test/Component Steps" on page 1819.

6. Set the data retrieval properties - optional

If you have data in the Data Pane, set the data retrieval properties. Click the **Link to a data source** button 🔗. For details, see the "Data Retrieval Options for Load Test Enabled Tests" on page 1845.

7. Set the run configuration to Release

In the General pane of the API Testing tab in the Options dialog (**Tools > Options > API Testing** tab **> General** node), in the **Run Sessions** options, select **Release**. The **Release** mode conserves resources, thus enhancing the load testing capabilities.

8. Run in Load Testing mode to validate the test

Expand the toolbar **Run** button and select **Run Test in Load Testing Mode**. This run is only for debugging purposes, to verify that the test is functional.

> **Note:** When you run a test in Load Testing mode, the Output pane does not contain data and the Run Results Viewer does not open. To view the results, select **Run** to run the test in functional mode.

9. Incorporate the test into LoadRunner

Add the test to the LoadRunner Controller console to include it in a load test.

How to Validate an XML file

Relevant for: API testing only

This task describes how to validate the code in an XML string. The Validate XML activity evaluates an XML string and checks its compliance with an XSD schema.

Since the **Validate XML** activity reads a string, you need to specify the actual XML string—not just a file name. If the string is short, you can copy it into the **Value** column directly. For more complex XML content, you can read the XML file using a **Read from File** activity.

This task includes the following steps:

- "Provide the XML code to validate" below
- "Add a Validate XML activity" below
- "Connect the steps - if using Read from File" below
- "Specify an XSD file" on the next page
- "Set a checkpoint - optional " on the next page
- "Run the test" on the next page

1. Provide the XML code to validate

To provide the XML code, you can paste an XML directly into the **Value** column, or read the contents from the file using the **Read from File** activity.

To enter XML code directly, copy the XML string from the source document and paste it into the **Value** column.

To read the XML from a file:

a. Drag the **File System > Read from File** activity onto the canvas.

b. Click the Input/Checkpoints tab in the Properties pane. Browse to the XML file you want to validate.

2. Add a Validate XML activity

Drag the **XML > Validate XML** activity onto the canvas. If you used a **Read from File** step, drag this activity beneath the step.

3. Connect the steps - if using Read from File

If you used a **Read from File** step to obtain the XML, follow these steps. Otherwise, skip to step 4.

a. Click the Input/Checkpoints tab in the Properties pane. Click in the **Value** column of the **XML String** property and click the **Link to a data source** button 🔗.

b. In the Select Link Source dialog box, select the **Available steps** option. In the right pane, select the **Read from File** step's output property, **Content,** and click **OK**.

4. Specify an XSD file

a. Select the **Validate XML** step in the canvas. Click the Input/Checkpoints tab in the Properties pane, Browse to an **XSD file**.

b. To import the XSD file into the test's folder, set the **XSD file to test folder** option to true. This is useful if you plan on saving the test on ALM. Setting this option will make the XSD file available to anyone who loads the test.

5. Set a checkpoint - optional

If you want to verify the results:

a. In the Input/Checkpoints tab in the Properties pane, click in the Checkpoints section.

b. Select the **Valid** check box. Set the value to true to verify that the XML code complies with the XSD. Set the value to false to confirm that the XML code did not comply with the XSD.

c. To check for a specific error, set the **Valid** value to false, and specify the expected error in the **Error** row. You can link to a data source containing the expected value.

6. Run the test

Run the test. The Output tab and Run Results Viewer will display the results and indicate whether the XML is in compliance with the XSD and details about the errors.

How to Transform an XML File

Relevant for: API testing only

This task describes how to transform an .xml file to a different structure based on an .xslt file.

Since the Transform XML activity reads a string, you need to specify the actual XML string—not just a file name. If the string is short, you can copy it into the **Value** column directly. For more complex XML content, you can read the XML file using a **Read from File** activity.

This task includes the following steps:

- "Provide the XML code to transform" on the next page
- "Add a Transform XML activity" on the next page
- "Connect the steps - if using Read from File" on the next page
- "Specify an XSLT file" on the next page

- "Set a checkpoint - optional " below
- "Run the test" below

1. Provide the XML code to transform

To provide the XML code, you can paste an XML directly into the **Value** column, or read the contents from the file using the **Read from File** activity.

To enter XML code directly, copy the XML string from the source document and paste it into the **Value** column.

To read the XML from a file:

a. Drag the **File System > Read from File** activity onto the canvas.

b. Click the Input/Checkpoints tab in the Properties pane. Browse to the .xml file you want to transform.

2. Add a Transform XML activity

Drag the **XML > Transform XML** activity onto the canvas. If you used a **Read from File** step, drag this activity beneath the step.

3. Connect the steps - if using Read from File

If you used a **Read from File** step to obtain the XML, follow these steps. Otherwise, skip to the next step.

a. Click the Input/Checkpoints tab in the Properties pane. Click in the **Value** column of the **XML String** property and click the **Link to a data source** button 🔗 .

b. In the Select Link Source dialog box, select the **Available steps** option. In the right pane, select the **Read from File** step's output property, **Content,** and click **OK**.

4. Specify an XSLT file

a. Select the **Transform XML** step in the canvas. Click the **Input/Checkpoints** tab in the Properties pane. Browse to an .xslt file.

b. To import the .xslt file into the test's folder, set the **XSLT file to test folder** option to true. This is useful if you plan on saving the test on ALM. Setting this option will make the .xslt file available to anyone who loads the test.

5. Set a checkpoint - optional

If you want to verify the results, select the **Transformed XML** check box in the **Checkpoints** section and specify the expected result. You can link to a data source containing the expected value.

6. Run the test

Run the test. The Output tab and the Run Results Viewer will indicate whether the transform operation succeeded.

How to Compare XML Strings

Relevant for: API testing only

This task describes how to compare two XML strings and view the differences.

This task includes the following steps:

- "Add an Compare XMLs step to the canvas" below
- "Enter the original XML string" below
- "Enter the new XML string" below
- "Set the ignore options - optional" below
- "Save the settings - optional" below
- "Set a checkpoint - optional" on the next page
- "View the results" on the next page

1. ## Add an Compare XMLs step to the canvas

   Drag the **XML > Compare XMLs** activity onto the canvas.

2. ## Enter the original XML string

   Copy the original XML string onto the clipboard.

   Click the Input/Checkpoints tab in the Properties pane. Click the down arrow in the **Value** column of the XML string 1 property and paste the original XML string into the box.

3. ## Enter the new XML string

   Copy the modified XML onto the clipboard and paste it into the XML string 2 row.

4. ## Set the ignore options - optional

   a. To ignore any differences in the order of the elements, set **Ignore element order** to true.

   b. To ignore any differences in namespaces, set **Ignore namespaces** to true.

   c. To ignore differences in a specific node, expand the **Ignore XPaths** array and add one or more array elements with the XPath expression you want to ignore.

5. ## Save the settings - optional

   To save all of your ignore settings for future use, click the **Save options** button.

6. **Set a checkpoint - optional**

   To set a checkpoint which indicates whether or not there was a change, click the **Validate** check box in the Checkpoints area.

   - To verify that there was a change, set **Are Equal** to false.
   - To verify that there was no change, set **Are Equal** to true.

7. **View the results**

   a. Run the test. In the Run Results Viewer, expand the results.

   b. Select the **Compare XMLs** node in the results tree in the left pane.

   c. In the **Captured Data** pane, click the **Report File** link. The report shows both XML strings. The report's legend indicates the nature of the change: added, removed, changed, and so forth.

   d. Scroll down in the **Captured Data** pane for the XML Comparison Results table. This table lists the changes, showing the old and new element and attribute names with their values.

References

Standard Activity Types

Relevant for: API testing only

This section describes the Standard activity groups:

Flow Control Activities

Relevant for: API testing only

The following activities allow you to control the flow of the test using loops, conditions, breaks, and delays:

- **Condition**. Enables you to use a branching mechanism to determine alternate test flows based on the value of a property. For details, see "Condition Activity" below.
- **Loop**. A loop frame whose properties can be set independently of the Test Flow. The loop types are **For loop**, **Do While loop**, and **For Each loop**. For details, see "Loop Activity" on the next page.
- **Break.** Halts the test execution.
- **Continue.** Resumes the test execution after a break.
- **Sleep.** Pauses the test execution for a specified amount of time.
- **Wait**. Waits a specified amount of time for a trigger event before releasing a step for execution. For details, see "Wait Activity" on page 1653.

Condition Activity

Relevant for: API testing only

This activity enables you to branch your test flow depending on a conditional outcome.

The following properties are available for this activity:

Property	Description

Use condition	Instructs UFT to use a conditional outcome to proceed with the test flow. If the current state of the application matches the defined condition, then the condition proceeds with the test steps defined in the **Yes** branch. If the current state of the application does not match the defined condition, the test proceeds with the steps in the **No** branch. You define the following settings for the condition activity: • **Variable:** the process for which you want to determine the value • **Operator:** the basis for comparison for the variable. You can select one of the following: ▪ = ▪ != ▪ > ▪ >= ▪ < ▪ <= • **Value:** the expected value for the variable You can link the variable and value to data sources or other test steps. For details, see "How to Assign Data to API Test/Component Steps" on page 1819.
Use event	Instructs UFT to use the event handler specified in the **Events** tab ⚡ to determine on which conditional branch to continue to run the test. **Note:** You can create custom properties to use in the event handler in by clicking the **Add** button ➕.

Loop Activity

Relevant for: API testing only

This activity enables you to use additional test flow loops independently of the test flow. This also enables you to set individual data flow patterns for each loop independently of the main **Test Flow** data usage properties.

The following iteration settings are available for the Test Flow or other custom loops:

Property	Description
Do While Loop - Use condition	Repeats the loop as long as the specified condition is met: • **Variable.** The variable to evaluate. Use the Select Link Source button 🔗 to enter a data source expression. • **Operator.** A comparison operator relevant to the variable such as **=**, **!=**, **Contains**, **Starts** and **Regex**. • **Value.** The value with which to compare the result.
Do While Loop - Use event	Performs iterations until the event returns True. This mode is useful for complex conditions that can be defined in an event handler. When you select this option, the Input Properties grid opens. • Click ➕ to define properties. • Click ⚡ to open the **Events** view. Click **Create a default handler** in the **Handler** column. • Modify or add code to the event handler method that defines your condition.
For Each Loop	Performs an iteration for each element in the associated array or collection of objects. Data is selected using the **Select Link Source** button 🔗. **Note:** When you delete data from a data table, it continues to run an iteration for that row, with empty values. To remove an entire row of a data, click the row and select **Delete** from the shortcut menu (only with Excel installations).
For Loop	Performs the Test Flow or custom loop the number of times that you specify in the **Number of Iterations** box.

Wait Activity

Relevant for: API testing only

This activity enables you to temporarily pause the test flow for a specified amount of time.

When running asynchronous service calls as part of your test, the **Wait** activity waits a specified amount of time for a trigger event before releasing the associated step. This activity is used in conjunction with the **HTTP Receiver** steps and Web Service calls imported as server response steps. For details, see "Asynchronous Service Calls" on page 1927.

The following properties are available:

Property	Description
Timeout	The timeout in milliseconds. Negative values indicate that there is no timeout.
Start timeout after step	The time after step execution from when to begin measuring the timeout interval.
Action on timeout	The action to take when the timeout is reached without the completion events: **Fail** or **Continue**.
Completion event names	A list of the completion events that were configured in prior receiver steps.

Miscellaneous Activities

Relevant for: API testing only

This activity group includes the following activities:

- **Custom Code.** Enables you to program a step to execute your own activity. Use the **Add** button to define new input or output properties.

- **Set Test Variable.** Defines a global variable for your test.

  Note: This adds an additional test variable to the built-in variables included in every test.

- **Run Program.** Invokes an application on the current machine. For details, see "Run Program Activity" below.

- **End Program.** Closes an application that is running. For details, see "End Program Activity" on the next page.

- **Report Message.** Sends a custom message to the Run Results and/or the Output pane. For details, see "Report Message Activity" on page 1656.

Run Program Activity

Relevant for API testing only

This activity enables you to run a program from your computer as part of the test.

The following properties are available for this activity:

Input Properties

Property	Description
Target	The name of the program to run. **Tip:** You can click the **Browse for file** button … to navigate to the file.
Command line arguments	The arguments to enter in the Windows command line to run the program selected for the **Target** property.
Working directory	The file system location of the program defined in the **Target** property. **Note:** If you do not remember the exact path to the Target program, click the **Browse for file** button … to find the correct directory.
Run as user	The user with which to run the program specified in the **Target** property.
Password	The password of the user specified in the **Run as user** property.
Domain	The password to use when starting the application.
Timeout	The amount of time (in milliseconds) to wait for the application to exit.

Checkpoint Properties

Property	Description
Exit code	The value returned by the program specified in the **Target** property when the program closed. **Tip:** You can choose an operator to expand the possible values if the output of the program varies. Select the appropriate operator from the dropdown menu.
Process terminated	Specifies whether the program completed its run correctly or timed out. Select the appropriate value from the drop-down menu in the **Value** column.

End Program Activity

Relevant for API testing only

This activity enables you to stop and close a program that is running as part of your test. You should use this step in conjunction with a **Run Program** activity.

The following properties are available for this activity:

Input Properties

When entering the input properties, you can choose between ending the program by specifying the path to the program, by specifying the window to close, or by specifying the process ID of the program.

> **Note:** You can only choose one of the options to close the program.

Property	Description
Executable path	The path to file to close. **Note:** You can click the **Browse for file** button ... to navigate the file.
Window Title	Enables you to specify a specific program window to close. In order to close the window, you must define the following properties: • **Title:** The window title of the program to close • **Use RegEx:** Instructs UFT to treat the window title as a regular expression. Select true or false from the drop-down list to enable or disable this attribute.
Process ID	The Windows process ID of the program to close.

Checkpoint Properties

Property	Description
Exit Code	The value provided by the program upon its completion.

Report Message Activity

Relevant for: API testing only

This activity enables you to send a message to the Output pane during a test run or to the run results.

The following properties are available for this activity:

Property	Description

Status	The status to report in the message. By default, UFT includes statuses of Done, Pass, and Fail. To define other statuses, click the **Link to data source** button in the **Value** cell. **Note:** This status is different from the run status of the step displayed for each test step in the Result Details pane in the Run Results Viewer.
Message	The message to report. You can manually enter the message or link this message to a data source or other test steps. For details on linking this property value, see "How to Assign Data to API Test/Component Steps" on page 1819.
Destination	The place to report the message. You can report this message in the run results only or in the run results and the Output pane (displayed during test runs).

The message generated during the test run of this step, including the **Status** and **Message** property values, is displayed in the **Captured Data** pane in the Run Results Viewer. For details on the Captured Data pane, see the section on the Captured Data Pane in the *HP Run Results Viewer User Guide*.

String Manipulation Activities

Relevant for: API testing only

This activity group provides access to the following string related activities:

- **Concatenate Strings.** Joins two strings.
- **Replace String.** Replaces part of a string with an alternate string. You can provide an array of replacement strings for use with iterations. For details, see "Replace String Activity" below.
- **Substring.** Extracts the characters between two specified locations of a string. For details, see "Substring Activity" on the next page.
- **Trim String.** Removes whitespace characters from the beginning or end of a string. For details, see "Trim String Activity" on page 1659.

Replace String Activity

Relevant for: API testing only

This activity enables you to replace a specified string with another string.

The following properties are available for this activity:

Input Properties

Property	Description

Source string	The name of the string to replace.
Search and replace array	The array containing the strings to find and replace. You must provide the values for the following: • **Search string:** The string to replace. • **Replace String:** The string which replaces the string found in the **Search String** property. **Tip:** If you need to replace multiple parts of the source string, add an additional array by clicking the **Add** button ✚ in the **Search and Replace (array)** property row.
Case-sensitive	Indicates that UFT should search for the **Search String** property by locating the matching case for the string specified in the Search String value cell. Select True or False from the drop-down list to set this property.

Checkpoint Properties

Property	Description
Result	The expected result for the search and replace operation. **Note:** Make sure to include spaces where needed in both the **Search** and **Replace** strings as well as the **Result** output string or the checkpoint will fail.

Substring Activity

Relevant for: API testing only

This activity enables you to extract a section of a specified string for use elsewhere in your test.

The following properties are available for this activity:

Input Properties

Property	Description
Source string	The string from which you are extracting a section.
Substring start	The character position (numerically) in the string defined in the **Source string** property to begin extracting.
Substring length	The number of characters to extract from the string specified in the **Source string** property.

Checkpoint Properties

Property	Description
Result	The expected string that is extracted.

Trim String Activity

Relevant for: API testing only

This activity enables you to trim extra whitespaces from a selected string.

The following properties are available for this activity:

Input Properties

Property	Description
Source string	The string from which to trim.
Trim start	Enables you to trim the whitespace from the beginning of the string. Choose True or False from the drop-down list to enable or disable trimming.
Trim end	Enables you to trim the whitespace from the end of the string. Choose True or False from the drop-down list to enable or disable trimming.

Checkpoint Properties

Property	Description
Result	The result of the trim operation. **Note:** Make sure to include spaces where needed in both the **Search** and **Replace** strings as well as the **Result** output string or the checkpoint will fail.

System Activities

Relevant for: API testing only

This activity group provides access to the following operating system activities:

- **Set OS Environment Variable.** Sets the value of an operating system variable.
- **Get OS Environment Variable.** Retrieves the value of an operating system variable.

Math Activities

Relevant for: API testing only

This activity group performs the following math related activities:

- **Add.** Adds two numbers.
- **Subtract.** Subtracts one number from another.
- **Multiply.** Multiplies two numbers.
- **Divide.** Divides one number by another.

Date and Time Activities

Relevant for: API testing only

This activity group performs the following date and time related activities:

- **Increment Date.** Adds date or time units to a date/time string. For details, see "Increment Date Activity" below.
- **Date/Time Difference.** Calculates the difference between two date/time strings. For details, see "Date/Time Difference Activity" on the next page.
- **Format Date/Time.** Formats a date/time string. For details, see "Format Date/Time Activity" on page 1662.

Increment Date Activity

Relevant for: API testing only

This activity enables you to change a date and time by a measured amount.

The following properties are available for this activity:

Input Properties

Property	Description
Original Date/Time	The source date and time to modify. **Tip:** You can manually enter the date and time or choose a date and time from the drop-down calendar.

Original Date/Time Format	The format of the date and time provided in the Original Date/Time property. Choose one of the formats from the drop-down list. **Note:** You only need to enter this information when you provide a string expression for the **Original Date/Time** property.
Units	The unit to add to the date and time specified in the **Original Date/Time** property. **Note:** You can only increment one unit at a time in a test step. If you need to increment multiple date and time units, add additional **Increment Date** activities to your test.
Units	The number of units (as specified in the **Units** property value) to add to the **Original Date/Time** property.

Checkpoint Properties

Property	Description
Result	The new date and time after the addition is made. **Note:** Make sure to enter the date and time with exactly the same format as specified in the **Original Date/Time Format** property.

Date/Time Difference Activity

Relevant for: API testing only

This activity enables you compare different dates and times.

The following properties are available for this activity:

Input Properties

Property	Description
Date/Time (A,B)	The date and times to compare. **Tip:** You can manually enter the date and time or choose a date and time from the drop-down calendar.

Date/Time Format (A,B)	The format of the date and time provided in the **Date/Time** property. Choose one of the formats from the drop-down list. **Note:** You only need to enter this information when you provide a string expression for the **Date/Time** property.
Units	The unit to compare between the date and times specified in the **Date/Time** properties. **Note:** You can only increment one unit at a time in a test step. If you need to increment multiple date and time units, add additional **Increment Date** activities to your test.

Checkpoint Properties

Property	Description
Result	The numeric difference between the compared dates specified in the **Date/Time** properties.

Format Date/Time Activity

Relevant for: API testing only

This activity enables you to format a selected date and time with a given date and time format.

The following properties are available for this activity:

Input Properties

Property	Description
Input Date/Time	The date and time to format. **Tip:** You can manually enter the date and time or choose a date and time from the drop-down calendar.
Input Date/Time Format	The format of the date and time specified in the Input Date/Time property. **Note:** You only need to enter this information when you provide a string expression for the **Input Date/Time** property.
Format	The format to which to change the date and time specified in the **Input Date/Time** property.

Checkpoint Properties

Property	Description
Result	The date and time after the new format is applied. **Note:** Make sure to enter the date and time with exactly the same format as specified in the **Format** property.

File System Activities

Relevant for: API testing only

This activity group provides access to the following file system related activities:

- **File Exists.** Searches for a specific file, or group of files, if you use a wildcard expression. It returns true if the file exists.
- **Folder Exists.** Searches for a specific folder. It returns true if the folder exists.
- **File Copy**. Copies a file to a new destination. For details, see "File Copy Activity" below.
- **File Move**. Moves a file to another destination. For details, see "File Move Activity" on the next page.
- **File Create**. Creates a new file. For details, see "Create File Activity" on page 1665.
- **File Delete.** Deletes a file.
- **File Rename**. Renames a file. For details, see "File Rename Activity" on page 1665.
- **Write to File**. Writes to an existing or For details, see "Write to File Activity" on page 1666.new file.
- **Read from File**. Retrieves text from a file. For details, see "Read From File Activity" on page 1667.
- **Get Folder Contents**. Gets the contents of a folder—file names, folder names, or both. For details, see "Get Folder Contents Activity" on page 1668.

File Copy Activity

Relevant for: API testing only

This activity enables you to copy a selected file to another specified location.

The following properties are available for this activity:

Input Properties

Property	Description
File source path	The path of the file to copy. **Tip:** You can click the **Browse for file** button ... to navigate to the file.
Target folder	The folder into which to copy the file specified in the File source path property. **Tip:** If you do not remember the exact path in which you want to copy the file, you can click the **Browse for folder** button ... to find the directory.
Target file name	The name of the file after it is copied.
Overwrite	Instructs UFT to overwrite or not overwrite an already existing file of the same name. Select True or False from the drop-down list to enable or disable the setting.

Checkpoint Properties

Property	Description
Result	Indicates whether the file copy succeeded or failed.

File Move Activity

Relevant for: API testing only

This activity enables you to move a file to a selected location.

The following properties are available for this activity:

Input Properties

Property	Description
File source path	The path of the file to copy. **Tip:** You can click the **Browse for file** button ... to navigate to the file.
Target folder	The folder into which to copy the file specified in the File source path property. **Tip:** If you do not remember the exact path in which you want to copy the file, you can click the **Browse for folder** button ... to find the directory.

Target file name	The name of the file after it is copied.
Overwrite	Instructs UFT to overwrite or not overwrite an already existing file of the same name. Select True or False from the drop-down list to enable or disable the setting.

Checkpoint Properties

Property	Description
Result	Indicates whether the file copy succeeded or failed.

Create File Activity

Relevant for: API testing only

This activity enables you to create a new file.

> **Note:** This activity creates a new file but does not put content into the file. If you need to also add content to the file, use a **Write to File** activity in conjunction with this activity.

The following properties are available for this activity:

Input Properties

Property	Description
Target folder	The folder into which to copy the file specified in the File source path property. **Tip:** If you do not remember the exact path in which you want to copy the file, you can click the **Browse for folder** button ... to find the directory.
Target file name	The name of the file after it is copied.
Overwrite	Instructs UFT to overwrite or not overwrite an already existing file of the same name. Select True or False from the drop-down list to enable or disable the setting.

Checkpoint Properties

Property	Description
Result	Indicates whether the file copy succeeded or failed.

File Rename Activity

Relevant for: API testing only

This activity enables you to rename a selected file.

The following properties are available for this activity:

Input Properties

Property	Description
File source path	The path of the file to copy. **Tip:** You can click the **Browse for file** button … to navigate to the file.
Target file name	The name of the file after it is copied.
Overwrite	Instructs UFT to overwrite or not overwrite an already existing file of the same name. Select True or False from the drop-down list to enable or disable the setting.

Checkpoint Properties

Property	Description
Result	Indicates whether the file copy succeeded or failed.

Write to File Activity

Relevant for: API testing only

This activity enables you to add content to a new or existing file.

You can use this activity in conjunction with the **Create File** activity to add content to a created file.

The following properties are available for this activity:

Input Properties

Property	Description
Content	The information to add to the file specified in the **File path** property. If you have a lot of content to add to the file, click the down arrow to display the extended
File path	The directory containing the file, including the file name.
Mode	The manner of editing the specified file. You can select one of the following: • **Append to existing file:** Adds content to a file that already exists. • **Create new file:** Creates a new file with the name specified in the **File path** property and with the content indicated in the **Content** property.

Encoding	The manner of encoding the content added in the file. Select one of the following modes of encoding: • ASCII • UTF-7 • UTF-8 • UTF-32 • Unicode
Append new line	Instructs UFT to add a new line for your content to the file selected (if you are writing to an existing file). Select True or False to enable or disable the setting.

Checkpoint Properties

Property	Description
Result	Indicates whether the file copy succeeded or failed.

Read From File Activity

Relevant for: API testing only

This activity enables you to check and read the content in a specific file.

The following properties are available for this activity:

Input Properties

Property	Description
File path	The directory containing the file, including the file name.
Encoding	The manner of encoding the content added in the file. Select one of the following modes of encoding: • ASCII • UTF-7 • UTF-8 • UTF-32 • Unicode

Checkpoint Properties

Property	Description
Content	The content read from the file.

Get Folder Contents Activity

Relevant for: API testing only

This activity enables you to check the contents of a specified folder.

The following properties are available for this activity:

Input Properties

Property	Description
Folder source path	The folder in which to search.
Search pattern	The criteria by which to search the folder's content. By default, UFT searches for any file having an extension (indicated by the *.* criteria.
Content type	The type of content for which to search. You can choose to search only files, only folders, or both files and folders.
Sort by	Sorts the folder content results. You can sort by a number of criteria: • **Name** • **Creation Time** (of the file/folder) • **Date Modified** (of the file/folder) • **None**
Sort type	Displays the results in ascending or descending order.
Include subfolders	Searches the subfolders included in the folder specified in the **Folder source path** property.

Checkpoint Properties

The checkpoint properties for this activity are displayed as an array, To add an array (or multiple arrays if needed), click the **Add Array** button ✚.

Property	Description
Name	The name of a file expected to be found.

Path	The directory of the expected result.
Full path	The full path to the directory of the expected result.
Content type	The expected content of the search.
Creation time	The creation time of the expected result.
Date modified	The modification date of the expected result.

Database Activities

Relevant for: API testing only

This activity group performs the following database related activities:

- **Open Connection.** Opens a connection to a database using the specified connection string. For details, see "Open Connection/Close Connection Activity" below.
- **Close Connection.** Closes an open database connection. For details, see "Open Connection/Close Connection Activity" below.
- **Select Data.** Executes a SELECT type SQL statement that retrieves data. For details, see "Select Data Activity" on the next page.
- **Execute Command.** Executes an SQL statement that does not retrieve data from the database, such as UPDATE, DELETE, and so forth. For details, see "Execute Command Activity" on page 1672.
- **Begin Transaction.** Begins a database transaction. For details, see "Begin Transaction Activity" on page 1673.
- **Commit Transaction.** Commits a database transaction. For details, see "Commit Transaction/Rollback Transaction Activity" on page 1674.
- **Rollback Transaction.** Rolls back a database transaction from a pending state. For details, see "Commit Transaction/Rollback Transaction Activity" on page 1674.

Open Connection/Close Connection Activity

Relevant for: API testing only

This activity enables you to open or close a connection to a database. You should use both of these activities together.

The **Open Connection** activity is required before other database connection activities.

The following properties are available for these activities:

Input Properties

Property	Description
Connection String (for Open Connection activities only)	An OLE DB database connection string. You can provide a value in one of the following ways: • Type it in manually. • Open the **Connection Builder** .... For details, see the "Connection Builder Dialog Box" on page 1720. • Use the **Link to a data source** button 🔗 to create a link to the string.
Connection (for Close Connection activities)	A link to the output of an **Open Connection** step. Use the **Link to a data source** button 🔗 to create a link.

Checkpoint Properties

Property	Description
Result	Indicates whether the connection is active (true) or closed (false).
Result Message	A message indicating success or an informational message from the database.

Select Data Activity

Relevant for: API testing only

This activity enables you to specify the data from a database user source.

You must put an **Open Connection** step in your test before this activity.

The following properties are available for this activity:

Input Properties

Property	Description
Connection	A link to the output of a previously opened connection string. Use the **Link to a data source** button 🔗 to create the link.

Property	Description
Query string	An SQL SELECT statement. You can provide a string in one of the following ways: - Type it in manually. - Open the **Query Builder** .... For details, see the "Query Builder Dialog Box" on page 1722. - Use the **Link to a data source** button 🔗 to create a link to the string.
Timeout	The maximum time to allow for executing the database command. A value of zero indicates no time limit. **Default:** 30 seconds

Checkpoint Properties

Property	Description
Count	The number of rows returned by the command.
Result	Indicates whether the operation has been successfully completed: true or false.
Result table	An array of returned rows. The row elements are column names. You can set an expected value for each column of each row: - The columns changed due to parameterization. - A column was removed since the original query. - You don't have access to database when designing the test. **Note:** In order to see the table column, you need to generate the output at least once. You can do this by selecting the **Generate output** option in the Query Builder, or by clicking the **Generate Output** button in the **Checkpoints** pane. **Tip:** Use the **Manage Columns** button to remove or replace columns when:

Database Property Options

Option	Description
Generate Output	Retrieves table data from the database and constructs an array with the current table structure. If you enabled **Generate output** in the Query Builder, you do not need to generate the output again. For details, see "Query Builder Dialog Box" on page 1722.
Manage Columns	If the table structure changed since you generated the output, the Manage Columns dialog box lets you manage the columns. This is common when columns names were a parameter, or if a column was deleted from the table.

Execute Command Activity

Relevant for: API testing only

This activity enables you to execute a command for your database data.

You must put an **Open Connection** step in your test before this activity.

The following properties are available for this activity:

Input Properties

Property	Description
Connection	A link to the output of a previously opened connection string. Use the **Link to a data source** button 🔗 to create the link.
Command	An SQL statement that does not retrieve data, such as UPDATE or DELETE. You can provide a string in one of the following ways: • Type it in manually in the drop-down edit box. • Use the **Link to a data source** button 🔗 to create a link to the string.
Timeout	The maximum time to allow for executing the database command. A value of zero indicates no time limit. **Default:** 30 seconds

Checkpoint Properties

Property	Description
Affected rows	The number of rows affected by the command.
Result	Indicates whether the command executed successfully: true or false.

Property	Description
Result Message	Success or upon failure, a message issued by the database indicating the nature of the failure.

Begin Transaction Activity

Relevant for: API testing only

This activity enables you to start a database transaction.

You must put an **Open Connection** step in your test before this activity.

The following properties are available for this activity:

Input Properties

Property	Description
Connection	A link to the output of a previously opened connection string. Use the **Link to a data source** button to create the link.
Isolation level	The isolation level of the transaction: - Default - Chaos - Read Committed - Read Uncommitted - Repeatable Read - Serializable - Snapshot. For details, see http://msdn.microsoft.com/en-us/library/system.data.isolationlevel.aspx.

Checkpoint Properties

Property	Description
Result	Indicates whether the connection is active (true) or closed (false).
Result Message	A message indicating success or an informational message from the database.

Commit Transaction/Rollback Transaction Activity

Relevant for: API testing only

This activity enables you to commit the results of a transaction to your database or roll back a selected transaction.

You must put a **Begin Transaction** step before this activity.

The following properties are available for this activity:

Input Properties

Property	Description
Transaction	A link to an open transaction. Use the **Link to a data source** button 🔗 to create a link to the output of a **Begin Transaction** step.

Checkpoint Properties

Property	Description
Result	Indicates whether the connection is active (true) or closed (false).
Result Message	A message indicating success or an informational message from the database.

FTP Activities

Relevant for: API testing only

This activity group provides activities that allow you to perform FTP operations, such as **FTP Upload** and **FTP Download**.

- **FTP Download.** Downloads a file or directory from an FTP server. For details, see "FTP Download Activity" on the next page.

- **FTP Upload.** Uploads a file or directory to an FTP server. For details, see "FTP Upload Activity" on page 1676.

- **FTP Rename.** Renames an existing file or directory on an FTP server. For details, see "FTP Rename Activity" on page 1678.

- **FTP File Delete.** Deletes a file on an FTP server. For details, see "FTP File Delete Activity" on page 1679.

- **FTP File Get Size.** Gets the size of a file on an FTP server. For details, see "FTP File Get Size Activity" on page 1680.

- **FTP Directory Delete.** Deletes a empty directory on an FTP server. For details, see "FTP Download Activity" below.

- **FTP Directory Create.** Creates a new directory on an FTP server. For details, see "FTP Directory Create Activity" on page 1681.

- **FTP Directory Get Content.** Lists the content of a directory on an FTP server. For details, see "FTP Directory Get Content Activity" on page 1683.

FTP Download Activity

Relevant for: API testing only

This activity enables you to download an FTP file from a specified directory.

The following properties are available for this activity:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.
Client certificate	• . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 1923. • A Choice element indicating the source of the client certificate for secure FTP connections: ▪ **Use file system certificate**: The expanded node provides the full path of the certificate file. ▪ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password.

Input Properties

Property	Description

URL	The URL location of the file to download.
User ID	The user name to access the FTP server.
Password	The password to access the FTP server.
Local Path	The location to which to download the file.
Overwrite	Enables UFT to overwrite or not overwrite an existing file by the same name. Choose True to overwrite a file of the same name or False to make another copy.
Transfer mode	The method of file transfer: ASCII or binary.

Checkpoint Properties

Property	Description
Result	An indicator if the step succeeded. Select True to add a checkpoint confirming the successful completion, or False to confirm an unsuccessful attempt to download the selected file.

FTP Upload Activity

Relevant for: API testing only

This activity enables you to upload a file to specified FTP directory.

The following properties are available for this activity:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Client certificate	• . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 1923. • A Choice element indicating the source of the client certificate for secure FTP connections: ▪ **Use file system certificate**: The expanded node provides the full path of the certificate file. ▪ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password.

Input Properties

Property	Description
Directory file path	The URL and directory in which to upload the file.
User ID	The user name to access the FTP server.
Password	The password to access the FTP server.
Local Path	The location from which to upload the file.
Overwrite	Enables UFT to overwrite or not overwrite an existing file by the same name. Choose True to overwrite a file of the same name or False to make another copy.
Transfer mode	The method of file transfer: ASCII or binary.

Checkpoint Properties

Property	Description
Result	An indicator if the step succeeded. Select True to add a checkpoint confirming the successful completion, or False to confirm an unsuccessful attempt to download the selected file.

FTP Rename Activity

Relevant for: API testing only

This activity enables you to rename a specified file in an FTP directory.

The following properties are available for this activity:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.
Client certificate	• . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 1923. • A Choice element indicating the source of the client certificate for secure FTP connections: ▪ **Use file system certificate**: The expanded node provides the full path of the certificate file. ▪ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password.

Input Properties

Property	Description
URL	The URL location of the file to rename.
User ID	The user name to access the FTP server.
Password	The password to access the FTP server.
New file name	The name of the file after renaming.

Checkpoint Properties

Property	Description
Result	An indicator if the step succeeded. Select True to add a checkpoint confirming the successful completion, or False to confirm an unsuccessful attempt to download the selected file.

FTP File Delete Activity

Relevant for: API testing only

This activity enables you to delete a selected file from an FTP directory.

The following properties are available for this activity:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.
Client certificate	• . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 1923. • A Choice element indicating the source of the client certificate for secure FTP connections: ▪ **Use file system certificate**: The expanded node provides the full path of the certificate file. ▪ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password.

Input Properties

Property	Description
URL	The URL location of the file to delete.

User ID	The user name to access the FTP server.
Password	The password to access the FTP server.

Checkpoint Properties

Property	Description
Result	An indicator if the step succeeded. Select True to add a checkpoint confirming the successful completion, or False to confirm an unsuccessful attempt to download the selected file.

FTP File Get Size Activity

Relevant for: API testing only

This activity enables you to check the size of a file on a specified FTP directory.

The following properties are available for this activity:

Input Properties

Property	Description
URL	The URL location of the file to check.
User ID	The user name to access the FTP server.
Password	The password to access the FTP server.

Checkpoint Properties

Property	Description
Result	An indicator if the step succeeded. Select True to add a checkpoint confirming the successful completion, or False to confirm an unsuccessful attempt to download the selected file.
File size	The expected size of the file.

FTP Directory Delete Activity

Relevant for: API testing only

This activity enables you to delete a selected FTP directory.

The following properties are available for this activity:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.
Client certificate	• . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 1923. • A Choice element indicating the source of the client certificate for secure FTP connections: ▪ **Use file system certificate**: The expanded node provides the full path of the certificate file. ▪ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password.

Input Properties

Property	Description
URL	The URL location of the file to download.
User ID	The user name to access the FTP server.
Password	The password to access the FTP server.

Checkpoint Properties

Property	Description
Result	An indicator if the step succeeded. Select True to add a checkpoint confirming the successful completion, or False to confirm an unsuccessful attempt to download the selected file.

FTP Directory Create Activity

Relevant for: API testing only

This activity enables you to create an FTP directory.

The following properties are available for this activity:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.
Client certificate	• . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 1923. • A Choice element indicating the source of the client certificate for secure FTP connections: ▪ **Use file system certificate**: The expanded node provides the full path of the certificate file. ▪ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password.

Input Properties

Property	Description
URL	The URL location of the directory to create.
User ID	The user name to access the FTP server.
Password	The password to access the FTP server.
Directory Path	The FTP location in which to create the new directory.

Checkpoint Properties

Property	Description
Result	An indicator if the step succeeded. Select True to add a checkpoint confirming the successful completion, or False to confirm an unsuccessful attempt to download the selected file.

FTP Directory Get Content Activity

Relevant for: API testing only

This activity enables you retrieve the content from a specified FTP directory.

The following properties are available for this activity:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.
Client certificate	• . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 1923. • A Choice element indicating the source of the client certificate for secure FTP connections: ▪ **Use file system certificate**: The expanded node provides the full path of the certificate file. ▪ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password.

Input Properties

Property	Description
URL	The URL location of the directory to check.
User ID	The user name to access the FTP server.
Password	The password to access the FTP server.

Checkpoint Properties

Property	Description

Result (array)	An array of possible property values for the content in the directory. Possible values include: • **Name** • **Type** • **Flags** • **Owner** • **Group** • **Size** • **Date Created**

Network Activities

Relevant for: API testing only

This activity group provides activities that enable you to send and receive HTTP and SOAP requests.

This activity group contains the following activities:

- **HTTP Request**. This activity enables you to send an HTTP request over the network. For details, see "HTTP Request Activity" below.
- **HTTP Receiver.** This activity enables you to receive an HTTP response from a server. For details, see "HTTP Receiver Activity" on page 1687.
- **SOAP Request**. This activity enables you to send a SOAP request over the network. For details, see "SOAP Request Activity" on page 1688.

HTTP Request Activity

Relevant for: API testing only

This activity enables you to add an HTTP request to a network-based service to your test.

The following properties are available for these activities:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

> **Note:** These properties are also relevant for Web Service request steps.

Property	Description
Proxy	The settings for the proxy server hosting the service contract: **Server** (URL and port when required), **Username**, and **Password**.
Authentication	The user credentials settings for obtaining a service contract: **Username** and **Password**.
Connection type	The type of connection: **Keep-Alive** or **Close**. **Default:** Keep-Alive
Timeout	The HTTP timeout in milliseconds.
Client certificate	A Choice element indicating the source of the client certificate: • **Use file system certificate**: The expanded node provides the full path of the certificate file. • **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. • **Password.** The certificate's password. **Note:** To use a certificate, make sure to the **Use Client Certificate** property to true.
Use Client Certificate	Enables the use of a client certificate. **Default:** false
Maximum automatic redirections	The number of times to attempt accessing a page, including redirection. **Default:** 3
Allow redirections	Indicates whether to allow the step to redirect to another URL. **Default:** true
Reuse cookies	Enables the reusing of cookies for the current step. **Default:** false.

Input Properties

Property	Description
URL	The URL to which to make the HTTP Request.

HTTP Method	The method by which to send the request. You can select one of the following: • GET • POST • PUT • DELETE • TRACE • OPTIONS • HEAD
HTTP Version	The HTTP version to use in the request. Choose from version 1.0 or version 1.1.
Request Headers	The request header to send with the HTTP request. This property is displayed as an array. You must provide the following details for the request: • **Name** • **Value** **Note:** To send multiple request headers, click the **Add Array Element** button ✚ in the **RequestHeaders** row.

Checkpoint Properties

Property	Description
HTTP Version	The HTTP version of the response to the HTTP request.
Status code	The expected code of the response.
Status description	The expected description of the response.
Response body as UTF-8 string	The expected response in UTF-8 form.
Response body as Base64 string	The expected response in Base64 form.

HTTP Receiver Activity

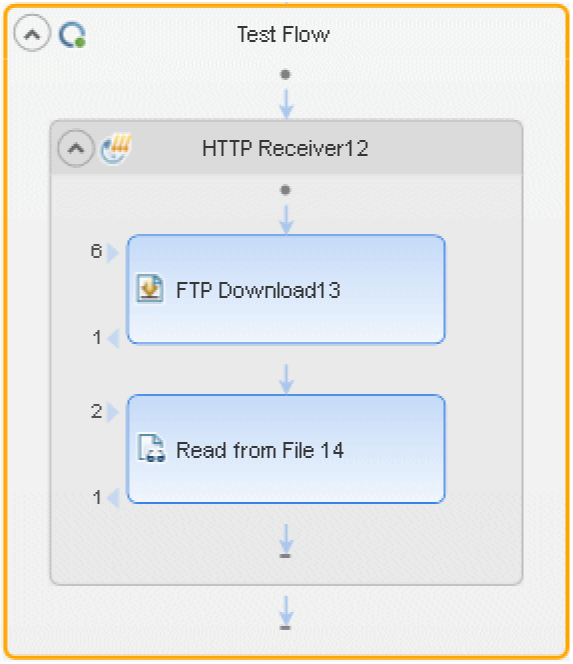
Relevant for: API testing only

The **HTTP Receiver** activity receives an HTTP message from a server. This activity is used with asynchronous messaging. For details, see "Asynchronous Service Calls" on page 1927.

Receiver activities are activities that act as receiver for a server response. This category includes the built-in **HTTP Receiver** activity and Web Service operations that were imported as a server response. For more details, see the "Select WSDL Dialog Box" on page 1764 or "Select WSDL Dialog Box" on page 1764.

All receiver activities are composite activities. This means that they serve as a wrapper for all activities inside the container whose responses are sent to the HTTP Receiver step—not the Test Flow.

For example, if an HTTP Receiver step contains the **FTP Download** and **Read from File** steps, these internal steps send their response to the HTTP Receiver step—not to the Test Flow loop.



The following properties are available for these activities:

General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Listen on port	The port upon which to listen for the server request.

Property	Description
Completion event name	The name of the event associated with a following **Wait** step, allowing the step to receive the server request.
Use SSL	Secures the connection with SSL: **true** or **false.**
SSL Certificate	Information about the SSL certificate: - **Subject.** The certificate's subject string. - **Store location.** The location of the certificate—LocalMachine or Windows store. - **Store name.** The name of the store hosting the certificate. - **Thumbprint.** The certificate's thumbprint or hash code.
Save response body with this extension	Saves the response body for messages with the specified extension.

Output Properties

You can link to the **HTTP Receiver** output property, its response body, through the Select Link Source dialog box. For details, see "Select Link Source Dialog Box (API Testing)" on page 1840.

Property	Description
Response body as a UFT-8 string	The body of the HTTP response in UFT-8 format.

SOAP Request Activity

Relevant for: API testing only

The **SOAP Request** activity sends a manual SOAP request to the server, usually in XML format. You can import a schema and load XML for this activity.

The following properties are available for these activities:

General Properties

To view the General properties, click the **General** button in the Properties pane.

> **Note:** These properties are also relevant for REST Service method steps.

Property	Description
Transport	The type of transport for the SOAP request: **HTTP** or **JMS**.

Property	Description
Transport > HTTP	Use the HTTP transport method for this call with the following properties: • **Endpoint Address.** The endpoint location of the service as derived from the WSDL • **SoapAction.** An expression indicating the SOAP action **Example:** HP.SOAQ.SampleApp/IHPFlights_Service/DeleteFlightOrder. • **ContentType.** The type of content: For example text/xml; charset=utf-8. • **Timeout.** The maximum time in milliseconds to wait for the response. Enter -1 to disable the timeout and wait for the response as long as required.
Transport > JMS	Use the JMS transport method for this call with the following properties: • **Send queue name.** The name of the queue from which to send the message. • **Receive queue name.** The name of the queue from which to receive the message. • **JMS send properties.** JMS properties with the key/value form. You can use the key to modify JMS header or message properties. • **JMS receive message selector.** An SQL expression to filter the received message. For more information on the syntax for defining selectors, see the **Message Properties** section in http://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html. For example, a key JMSCorrelationID, with a value of 4566636, receives only messages whose correlation ID is 4566636.
IsOneWay	Indicates whether the service is a one way service: true or false. A one way service only sends a request. A non-one way service sends a request and receives a response.

The input and checkpoint properties for the SOAP Request activity are dependent on the file you load containing your Request and Response information.

JSON Activities

Relevant for: API testing only

This activity group contains activities related to JSON files:

- **JSON to String.** Converts a JSON file to a text string. For details, see "JSON to String Activity" below.
- **String to JSON.** Converts a string to a JSON structure. For details, see "String to JSON Activity" below.

This **JSON to String** and **String to JSON** activities are useful when you need to access the output in a specific format—either as string or JSON.

For example, suppose your test contains a Web Service call whose output is JSON. However, you need to use the output as an input for the next step, such as an API test step, which requires a textual string—not JSON.

You can use the **JSON to String** activity to create a string that is compatible with the API test step.

After the API test step, you can use a **String to JSON** activity to convert the output to JSON, making it available for linking and checkpoints.

String to JSON Activity

Relevant for: API testing only

This activity enables you to convert a selected string to JSON format.

The following properties are available for this activity:

Input Properties

Property	Description
Source string	The string to convert into JSON text.

The checkpoints are dependent upon the JSON text contained in the file you load in the checkpoints section.

JSON to String Activity

Relevant for: API testing only

This activity enables you to take JSON text and convert it into a string.

The following properties are available for this activity:

The input properties are dependent upon the JSON text contained in the file you load in the checkpoints section.

Checkpoint Properties

Property	Description
Source string	The string to convert into JSON text.

Java Activities

Relevant for: API testing only

This activity group contains the **Call Java Class** activity. It enables you to set up and configure a call to a Java class.

The input and checkpoint properties for this activity are dependent on the information in your Java class file.

JMS Activities

Relevant for: API testing only

The JMS transport method is a J2EE standard for sending messages—either text or Java objects—between Java clients. UFT supports the sending of text messages between Java clients. There are two scenarios for communication:

- **Peer-to-Peer.** Also known as **Point-to-Point**. JMS implements point-to-point messaging by defining a message queue as the target for a message. Multiple senders send messages to a message queue, and the receiver gets the message from the queue.

- **Publish-Subscribe.** Each message is sent from one publisher to many subscribers through a designated topic. The subscribers only receive messages sent after they have subscribed.

To interpret Topic and Queue names, UFT calls a lookup method on a JNDI context, defined in the Properties pane's Test Settings. For details, see the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

For task details, see "How to Retrieve Messages from a JMS Queue" on page 1630 or "How to Receive Messages Through JMS Topics" on page 1633.

This activity group provides steps that handle JMS messages, and contains the following activities:

- **Publish Message to JMS Topic.** Publishes a message to multiple subscribers using a JMS topic. For details, see "Publish Message to JMS Topic Activity" on the next page.

- **Receive Message from JMS Queue.** Receives a message from a JMS queue. For details, see "Send Message to JMS Queue Activity" on page 1695.

- **Receive Message from JMS Topic.** Receives published messages to a specific JMS topic for a subscription. Place this step after **Subscribe to JMS Topic**. For details, see "Receive Message from JMS Topic Activity" on page 1694.

- **Send Message to JMS Queue.** Sends a message to a JMS queue. For details, see "Send Message to JMS Queue Activity" on page 1695.
- **Send and Receive Message from JMS Queue.** Sends a message to a specified queue and receives a message from a specified queue. For details, see "Send/Receive Messages from JMS Queue Activity" on page 1695.
- **Subscribe to JMS Topic.** Creates a subscription for a JMS topic. Use this step before any **Receive Message from Topic** steps for the subscription. For details, see "Subscribe to JMS Topic Activity" on page 1697.
- **Browse JMS Queue Messages.** Retrieves messages from the JMS queue without consuming them. For details, see "Browse JMS Queue Activity" on page 1697.

Publish Message to JMS Topic Activity

Relevant for: API testing only

This activity enables you to publish a message to a specified JMS topic.

The following properties are available for these activities:

Input Properties

Property	Description
Topic name	The name of the queue from which to receive the message.
Message	The message to publish.
JMS send properties	JMS properties with the key/value form. You can use the key to modify JMS header or message properties.

Checkpoint Properties

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Receive Message from JMS Queue Activity

Relevant for: API testing only

This activity enables you receive a message from a selected JMS Queue.

The following properties are available for this activity:

Input Properties

Property	Description
JMS receive message selector	An SQL expression to filter the received message. For details on the syntax for defining selectors, see the **Message Properties** section in http://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html. For example, a key JMSCorrelationID, with a value of 4566636, receives only messages whose correlation ID is 4566636.
Receive queue name	The name of the queue from which to receive the message. If the value is empty, it uses a temporary queue.

Checkpoint Properties

Property	Description
JMSMessageID	A unique ID for the message.
JMSTimeStamp	The time and date of the message.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSReplyTo	The contents of the **Reply To** field.
JMSDestination	The message's destination.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. False by default.
JMSType	A string describing the JMS type.
JMSExpiration	The message's expiration date.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMS properties	An array of message properties in the structure of keys and values. **Tip:** Use the **Number of elements** drop -down to specify the number of properties, or click the **Add** button to add elements.
Message body	The message content.

Receive Message from JMS Topic Activity

Relevant for: API testing only

This activity enables you to receive a message from a selected JMS topic.

The following properties are available for this activity:

Input Properties

Property	Description
Topic name	The name of the topic from which to receive the message.
Subscription name	The name of the subscription. Use the same subscriber name value from the previously-called **Subscribe to JMS Topic** step.

Checkpoint Properties

Property	Description
JMSMessageID	A unique ID for the message.
JMSTimeStamp	The time and date of the message.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSReplyTo	The contents of the **Reply To** field.
JMSDestination	The message's destination.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. False by default.
JMSType	A string describing the JMS type.
JMSExpiration	The message's expiration date.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMS properties	An array of message properties in the structure of keys and values. **Tip:** Use the **Number of elements** drop -down to specify the number of properties, or click the **Add** button to add elements.
Message body	The message content.

Send Message to JMS Queue Activity

Relevant for: API testing only

This activity enables you to send a message to a selected JMS queue.

The following properties are available for this activity:

Input Properties

Property	Description
Send queue name	The name of the queue from which to send the message.
Message	The message to send.
JMS send properties	JMS properties with the key/value form. You can use the key to modify JMS header or message properties.

Checkpoint Properties

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Send/Receive Messages from JMS Queue Activity

Relevant for: API testing only

This activity enables you to send or receive messages from a selected JMS queue activity.

The following properties are available for this activity:

Input Properties

Property	Description
Send queue name	The name of the queue to which to send the message.
Receive queue name	The name of the queue from which to receive the message. If the value is empty, it uses a temporary queue.
Message	The message to send.
JMS send properties	JMS properties with the key/value form. You can use the key to modify JMS header or message properties.

Property	Description
Send queue name	The name of the queue to which to send the message.
JMS receive message selector	An SQL expression to filter the received message. For details on the syntax for defining selectors, see the **Message Properties** section in http://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html. For example, a key JMSCorrelationID, with a value of 4566636**,** receives only messages whose correlation ID is 4566636. **Tip:** To automatically generate a selector, use the **Automatically generate selector** option in the Properties pane's **Test Settings** tab. This selector uses a correlation ID that corresponds to the request's message ID.

Checkpoint Properties

Property	Description
JMSMessageID	A unique ID for the message.
JMSTimeStamp	The time and date of the message.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSReplyTo	The contents of the **Reply To** field.
JMSDestination	The message's destination.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. False by default.
JMSType	A string describing the JMS type.
JMSExpiration	The message's expiration date.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMS properties	An array of message properties in the structure of keys and values. **Tip:** Use the **Number of elements** drop -down to specify the number of properties, or click the **Add** button to add elements.
Message body	The message content.

Subscribe to JMS Topic Activity

Relevant for: API testing only

This activity enables you to subscribe to a selected JMS topic.

The following properties are available for this activity:

Input Properties

Property	Description
Topic name	The name of the topic from which to receive the message.
Subscription name	The name of the subscription. To retrieve messages on the subscription created by this step, insert a **Receive Message from Topic** step using the same **Subscription name** value.
JMS receive message selector	An SQL expression to filter the received message. For details on the syntax for defining selectors, see the **Message Properties** section in http://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html.

Checkpoint Properties

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Browse JMS Queue Activity

Relevant for: API testing only

This activity enables you to browse a selected JMS queue.

The following properties are available for this activity:

Input Properties

Property	Description
Queue name	The name of the queue upon which to browse for a collection of messages. This field is mandatory. **Note:** When using IBM ActiveMQ, you must use the format dynamicQueues/<queue_name>. If the specified queue does not exist, it will be created dynamically.
Messages selector	An optional SQL expression to filter the received messages. For details on the syntax for defining selectors, see the **Message Properties** section in http://www.oracle.com/technetwork/java/jms-101-spec-150080.pdf.

Checkpoint Properties

Property	Description
JMSMessageID	A unique ID for the message.
JMSTimeStamp	The time and date of the message.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSReplyTo	The contents of the **Reply To** field.
JMSDestination	The message's destination.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. False by default.
JMSType	A string describing the JMS type.
JMSExpiration	The message's expiration date.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMS properties	An array of message properties in the structure of keys and values. **Tip:** Use the **Number of elements** drop -down to specify the number of properties, or click the **Add** button to add elements.
Message body	The message content.

Load Testing Activities

Relevant for: API testing only

Load testing enable you to measure the performance of your application or service with a defined user load.

When you have a full LoadRunner installation on the same machine, UFT provides a custom template, **API Load Test**.

For tests created with the standard **API Test** template, UFT provides a conversion utility which converts the script into a LoadRunner compatible script, with a **.usr** extension.

> **Note:** Tests created in UFT and converted into LoadRunner scripts, cannot be edited in LoadRunner's Virtual User Generator (VuGen). To edit a test, modify it in UFT.

When working with data tables, you can use the standard data retrieval methods that are available in LoadRunner. For details, see "Data Retrieval Options for Load Test Enabled Tests" on page 1845.

For task details, see "How to Prepare and Run a Load Test" on page 1642.

This activity group contains the following activities that allow you prepare the test for load testing:

- **Start Transaction.** Marks the beginning of a LoadRunner transaction.
- **End Transaction.** Marks the end of a LoadRunner transaction.
- **Think Time.** Emulates a time delay, in seconds, between steps.

For details, see "Prepare for load testing - optional" on page 1643.

IBM WebSphere MQ Activities

Relevant for: API testing only

This activity group enables you to perform actions on an IBM Websphere MQ application server.

The activities in this group are:

- **Connect to MQ Queue Manager.** Connects to a specific MQ Queue Manager and maintains an open connection with the server. For details, see "Connect to MQ Queue Manager Activity" on the next page.
- **Disconnect from MQ Queue Manager.** Disconnects from the specific MQ Queue Manager and closes the connection with the MQ server. For details, see "Disconnect from MQ Queue Manager Activity" on page 1701.
- **Commit MQ Pending Messages.** Commits the last changes made for all messages since the last point of synchronization. For details, see "Commit/Backout Pending MQ Messages Activity" on page 1702.
- **Backout MQ Pending Messages.** Performs a Backout operation from the last point of synchronization. All PUT messages will be cancelled and GET messages will be reinstated to the queue. For details, see "Commit/Backout Pending MQ Messages Activity" on page 1702.
- **Browse Messages in MQ Queue.** Browses the messages on the MQ Queue via the Queue Manager. For details, see "Browse Messages in MQ Queue Activity" on page 1702.
- **Put Message to MQ Queue.** Puts a message on the MQ queue via the Queue Manager. For details, see "Put Message to MQ Queue Activity" on page 1704.
- **Get Message from MQ Queue.** Gets the most recent message from the MQ queue via the Queue Manager. For details, see "Get Message from MQ Queue Activity" on page 1705.
- **Put and Get Message from MQ Queue.** Puts a message on the MQ queue and then gets a message from the MQ queue via the Queue Manager. For details, see "Put and Get Message

from MQ Queue" on page 1707.

- **Subscribe to MQ Topic.** Creates a subscription for an MQ topic. From this point on, a subscriber can receive messages from a specified topic. For details, see "Subscribe to MQ Topic Activity" on page 1709.
- **Unsubscribe to MQ Topic.** Cancels a subscription for the specified MQ topic. For details, see "Unsubscribe from MQ Topic Activity" on page 1710.
- **Publish Message to MQ Topic.** Publishes a message to subscribers using an MQ topic. For details, see "Publish Message to MQ Topic Activity" on page 1711.
- **Receive Message from MQ Topic.** Receives messages for the active subscription, that were published to a specific MQ topic. For details, see "Receive Message from MQ Topic Activity" on page 1712.

UFT enables you to set the data format of the message for both queues and topics. The **String** format indicates Unicode and non- Unicode strings without a prefix. The **UTF** format represents a UTF string with a 2-byte prefix.

For task details, see "How to Retrieve Messages from an MQ Queue" on page 1635 or "How to Receive Messages Through MQ Topics" on page 1636.

Connect to MQ Queue Manager Activity

Relevant for: API testing only

This activity enables you to connect to the IBM Websphere MQ Queue Manager.

The properties available for this activity include:

Input Properties

Property	Description
Host name	The name or IP address of the Queue Manager's host machine.
Port	The port through which to connect, by default **1414**.
Channel	The channel through which to connect. For most configurations, use SYSTEM.DEF.SVRCONN.
Queue manager name	An property indicating the name of the Queue Manager.
User ID	An optional identification property for connecting to the host machine.
Password	An optional password for connecting to the host machine.

Property	Description
Host name	The name or IP address of the Queue Manager's host machine.
Port	The port through which to connect, by default **1414**.
SSL	Enables or disables SSL. When you enable SSL, you can set values for the following properties: • **SSLCipherSpec.** The SSL Cipher Specification, SSLCIPH. This specification indicates which data encryption algorithm and key size to use, such as DES_SHA_EXPORT or RC2_MD5_EXPORT. • **SSLKeyRepository.** The path of the SSL key repository on the client machine.

Checkpoint Properties

Property	Description
MQManager	An automatically generated connection expression. **Note:** • Although this does not appear in the **Checkpoints** grid, it is visible as an output property in the Select Link Source dialog box. For details, see the "Select Link Source Dialog Box (API Testing)" on page 1840. • When you add steps, they automatically use the connection expression from the most recent connection step.
Result	A boolean variable indicating whether the step succeeded.

Disconnect from MQ Queue Manager Activity

Relevant for: API testing only

This activity enables you to disconnect from the IBM Websphere MQ Queue Manager.

The properties available for this activity include:

Input Properties

Property	Description
MQManager	A connection expression linked to a **Connect to MQ Queue Manager** step. **Note:** When you add a **Disconnect to MQ Queue Manager** step, it automatically creates a link to the most recent connection step.

Checkpoint Properties

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Commit/Backout Pending MQ Messages Activity

Relevant for: API testing only

This activity enables you to commit or roll back pending messages from a MQ Queue.

You must put a **Connect to MQ Queue Manager** step before this step.

The properties available for this activity include:

Input Properties

Property	Description
MQManager	A connection expression linked to a **Connect to MQ Queue Manager** step. **Note:** When you add this step, it automatically creates a link to the most recent connection step.

Checkpoint Properties

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Browse Messages in MQ Queue Activity

Relevant for: API testing only

This activity enables you to browse all the messages in a selected MQ Queue.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Queue name	The name of the MQ queue to browse.
Queue Access Options	A tree with several built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Message Get Options	A tree with several built-in MQ Get options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • -1: Unlimited wait time • 0: No wait • A positive number: The time to wait in milliseconds.
Message data format	The format of the message being retrieved: **String** or **UTF**.
Match Conditions	An array of match conditions in a key/value format. Select a key from the drop down box in the Key's **Value** column and provide a value. MQMO_NONE does not require a value.

Checkpoint Properties

Property	Description
JMSMessageID	A unique ID for the message.
JMSTimeStamp	The time and date of the message.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.

Property	Description
JMSMessageID	A unique ID for the message.
JMSReplyTo	The contents of the **Reply To** field.
JMSDestination	The message's destination.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. False by default.
JMSType	A string describing the JMS type.
JMSExpiration	The message's expiration date.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMS properties	An array of message properties in the structure of keys and values. **Tip:** Use the **Number of elements** drop -down to specify the number of properties, or click the **Add** button to add elements.
Message body	The message content.

Put Message to MQ Queue Activity

Relevant for: API testing only

This activity enables you to add a message to a selected MQ Queue.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Put queue name	The name of the MQ queue on which to put the message.
Queue Access Options	A drop down of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.

Property	Description
Message Put Options	A drop down of built-in message put options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
Correlation ID	The correlation ID of the message.
Message format	The format of the message being sent: **String** or **UTF**.
Character Set	The character set code to use when sending a message. **Note:** This property is only relevant when the message format is set to String.
Message body	The body of the message to put on the queue.
Message Properties	An array of Put message properties in a Key/Value format.

Checkpoint Properties

Property	Description
MessageId	A message identifier of the message being sent.
PutTime	The date and time the message was put.
UserId	The sender's user ID.

Get Message from MQ Queue Activity

Relevant for: API testing only

This activity enables you to receive a message from a selected MQ Queue.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Get Queue Name	The name of the MQ queue from which to get the message.
Queue Access Options	A drop down of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Message Get Options	A collection of built-in Message get options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • -1: Unlimited wait time • 0: No wait • A positive number: The time to wait in milliseconds.
Message data format	The format of the message being retrieved: **String** or **UTF**.
Match Condition	An array of match conditions in a key/value format. Select a key from the drop down box in the Key's **Value** column and provide a value. MQMO_NONE does not require a value.

Checkpoint Properties

Property	Description
MessageId	The message identifier of the retrieved message.
CorrelationId	The correlation identifier of the retrieved message.
GroupId	An identifier of the message group to which the physical message belongs.
MsgSeqNumber	The sequence number of a logical message within a group.

Property	Description
MessageId	The message identifier of the retrieved message.
PutTime	The date and time the PUT action was executed for the message.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
UserId	The User ID of the user who submitted the message.
MessageBody	A string representing the UTF content of the retrieved message.

Put and Get Message from MQ Queue

Relevant for: API testing only

This activity enables you to add and receive a message from a selected MQ Queue.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Auto Match Correlation ID	Correlates the sent and received messages by automatically adding a match condition. This condition indicates that the **Correlation ID** of the received message should equal the **Message ID** of the sent message.
Put Queue Name	The name of the MQ queue upon which to put the message.
Put Queue Access Options	A collection of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Message Put Options	A collection of built-in message put options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Put priority	The priority of the message to be submitted.

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Put correlation ID	A correlation identifier for the message to be submitted.
Put Message Format	The format of the message: being sent: **String** or **UTF**.
Character Set	
Message Body	A string representing the UTF content of the submitted message.
Message properties	An array of Put message properties in a key/value format.
Get Queue Name	The name of the MQ queue from which to get the message.
Get Queue Access Options	A collection of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Message Get Options	A collection of built-in options that control the Get operation. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • -1: Unlimited wait time • 0: No wait • A positive number: The time to wait in milliseconds.
Get Message Format	The format of the received message: **String** or **UTF**.

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Match Condition	An array of Get Message match conditions n a key/value format. Select a key from the drop down box in the Key's **Value** column and provide a value. MQMO_NONE does not require a value.

Checkpoint Properties

Property	Description
MessageId	The message identifier of the retrieved message.
CorrelationId	The correlation identifier of the retrieved message.
GroupId	An identifier of the message group to which the physical message belongs.
MsgSeqNumber	The sequence number of a logical message within a group.
PutTime	The date and time the PUT action was executed for the message.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
UserId	The User ID of the user who submitted the message.
MessageBody	A string representing the UTF content of the retrieved message.

Subscribe to MQ Topic Activity

Relevant for: API testing only

This activity enables you to subscribe to a selected MQ Queue topic.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Topic name	The name of the MQ topic from which subscribers can access the message.
Subscription name	An automatically generated subscription name.

Property	Description
Topic Access Options	A collection of built-in MQ topic access options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.

Checkpoint Properties

Property	Description
Result	The Subscription name entered manually or generated automatically. **Notes:** • Although this does not appear in the **Checkpoints** grid, it is visible as an output property in the "Select Link Source Dialog Box (API Testing)". • This property can be used as an input value for the subscription name in the **Unsubscribe from MQ Topic** and **Receive Message from MQ Topic** steps.

Unsubscribe from MQ Topic Activity

Relevant for: API testing only

This activity enables you to unsubscribe from a previous subscription to an MQ Queue topic.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Subscription name	The subscription from which to unsubscribe. This can be a literal string or an expression linked to the output of a **Subscribe to MQ Topic** step.

Checkpoint Properties

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Publish Message to MQ Topic Activity

Relevant for: API testing only

This activity enables you to publish a message to a selected MQ Queue topic.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Topic Name	The name of the topic on which to publish the message.
Topic Access Options	A drop down of the built-in options that control the opening of the topic. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Message Put Options	A drop down of built-in options that control the Put operation, such as MQPMO_SYNCPOINT. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
Correlation ID	The UserId of the user who submitted the message.
Message Format	The format of the message being published: **String** or **UTF**.
Character Set	The character set to use when sending this message. **Note:** This property is only relevant when the Message Format is set to String.
Message body	The message to publish.

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Topic Name	The name of the topic on which to publish the message.
Message Properties	An array of message properties in a Key/Value format.

Checkpoint Properties

Property	Description
MessageId	A message identifier of the submitted message.
PutTime	The date and time the message was put.
UserId	The UserId of the user who submitted the message.

Receive Message from MQ Topic Activity

Relevant for: API testing only

This activity enables you to receive a message from a selected MQ Queue topic.

You must put a **Connect to MQ Queue Manager** step before this step.

The following properties are available for this activity:

Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Subscription Name	The subscription through which to receive topic messages. This can be a literal string or an expression linked to the output of a **Subscribe to MQ Topic** step.
Message Get Options	A drop down of built-in message get options. **Note:** To access additional options, use an event handler. For details, see "Writing Event Handlers for API Test Steps" on page 1935.

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Subscription Name	The subscription through which to receive topic messages. This can be a literal string or an expression linked to the output of a **Subscribe to MQ Topic** step.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • -1: Unlimited wait time • 0: No wait • A positive number: The time to wait in milliseconds.
Message Format	The format of the message being retrieved: **String** or **UTF**.
Match Condition	An array of match conditions n a key/value format. Select a key from the drop down box in the Key's **Value** column and provide a value. MQMO_NONE does not require a value.

Checkpoint Properties

Property	Description
MessageId	The message identifier of the retrieved message.
CorrellationId	The correlation identifier of the retrieved message.
GroupId	An identifier of the message group to which the physical message belongs.
MsgSeqNumber	The sequence number of a logical message within a group.
PutTime	The date and time the message was put.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
UserId	The User ID of the user who submitted the message.
MessageBody	A string representing the UTF content of the retrieved message.

HP Automated Testing Tools Activities

Relevant for: API testing only

This activity group contains activities that allow you implement unified testing plan. For task details, see "How to Call External Tests or Actions" on page 1639.

- **Call API Action or Test.** Calls an API or Service Test test or action when the current Test Flow or loop reaches this step.
- **Call GUI Action or Test.** Calls a GUI test or action when the current Test Flow or loop reaches this step. Do not insert a call to a GUI Test action or test that contains a call to a API Test action or test, as this can cause unexpected behavior.
- **Call Virtual User Generator script.** Runs a LoadRunner Virtual User Generator (VuGen) script.

General Properties

The following general properties are available for these activities:

Property	Activity	Description
Test path	**Call API Action or Test** **Call GUI Action or Test**	
Action name	**Call API Action or Test** **Call GUI Action or Test**	The name of the action being called. This field is read-only. To select an action, click the **Select Action or Test** button in the **Input/Checkpoints** tab.
Description	**All**	An editable field describing the call.
Result Directory	**Call API Action or Test**	The location in which to store the run results. This field is editable.
Script path	**Call Virtual User Generator Script**	The path of the Virtual User Generator-LoadRunner script. This field is read-only. To select a script, click the **Select Virtual User Generator Script** button in the Input/Checkpoint view.

SAP Activities

Relevant for: API testing only

This activity group contains the activity related to SAP IDoc status.

- **Get IDOC Status.** Retrieves the status of an IDoc.

This section describes the built-in SAP activity. For information about imported SAP activities representing RFCs and IDocs, see "How to Create an SAP API Test Step" on page 1638.

General Properties

To view these properties, click the **General** tab in the Properties pane.

Property	Description
Timeout	The maximum time allowed for the function to respond in milliseconds. **Default:**100000
ConnectionInfo	The SAP Connection information for the selected step: • **Server.** The name or IP address of the server. • **System number.** The server's System Number. • **Client.** The client for this SAP connection. • **Username.** The username for this SAP connection. • **Password.** The password for this SAP connection.
Function name	The name of the RFC (for RFCs only).
Expect Exception	Indicates whether or not to expect an exception when running the function: true or false (for RFCs only).
IDocController	IDoc Controller properties (for IDocs only): • **Message Type**, **Basic Type**, and **Extension**. • **Sender:**Port, Partner number, and Partner type. • **Receiver:**Port, Partner number, and Partner type.
Transaction	Indicates whether or not the RFC is part of a transaction or not: true or false (for RFCs only).

Input Properties

Property	Description
IDoc Number	The IDoc number for the IDoc whose status you want to retrieve.

Checkpoint Properties

The checkpoint name corresponds to the SAP property ID.

Property	Description
MANDT	The client.
DOCNUM	The IDoc number.
LOGDAT	The date of the status information.
LOGTIM	The time of the status information.
COUNTR	The IDoc status counter.
CREDAT	The creation date of the status record.
CRETIM	The creation time of the status record.
STATUS	The status of the IDoc.
UNAME	The user name.
REPID	The program name.
ROUTID	The name of the subroutine or function module.
STACOD	The status code.
STATXT	The status code textual representation.
SEGNUM	The SAP segment number.
SEGFLD	The field name in the SAP segment.
STAPA1,2,3,4	Status parameters 1, 2, 3, and 4.
STATYP	The system message type: A, W, E. S, or I.
STAMQU	The status message qualifier.
STAMID	The status message ID.
STAMNO	The status message number.
TID	The transaction ID.
APPL_LOG	The application log.

XML Activities

Relevant for: API testing only

This activity group contains activities related to XML files:

- **Transform XML.** Converts an XML file to another structure based on the specified XSLT. For details, see "Transform XML Activity" below.
- **Compare XML.** Performs a comparison between two XML strings. For details, see "Compare XML Activity" on the next page.
- **XML to String.** Converts an XML file to a text string. For details, see "XML to String Activity" on page 1719.
- **String to XML.** Converts a string to XML structure. For details, see "String to XML Activity" on page 1719.
- **Validate XML.** Validates an XML file against an XSD schema. For details, see "Validate XML Activity" on page 1719.

The **XML to String** and **String to XML** activities are useful when you need to access the output in a specific format—either as string or XML.

For example, suppose your test contains a Web Service call whose output is XML. However, you need to use the output as an input for the next step, which requires a textual string—not XML.

You can use the **XML to String** activity to create a string that is compatible with the API test step.

After the test step, you can use a **String to XML** activity to convert the output back to XML, making it available for linking, and checkpoints.

For task information, see:

- "How to Validate an XML file" on page 1645
- "How to Transform an XML File" on page 1646
- "How to Compare XML Strings" on page 1648

Transform XML Activity

Relevant for: API testing only

This activity enables you to transform a selected XML string.

The following properties are available for this activity:

Input Properties

Property	Description

XML String	The XML to transform.
Import XSLT to test folder	Copies the file containing the transformed XML (the .xslt file) to the test folder.
XSLT folder	The full path of the folder in which to place the file containing the transformed XML,

Checkpoint Properties

Property	Description
Transformed XML	The expected XML string after its transformation.

Compare XML Activity

Relevant for: API testing only

This activity enables you to compare two selected XML strings.

The following properties are available for this activity:

Input Properties

Property	Description
XML String 1	The first XML string to compare.
XML String 2	The second XML string to compare
Ignore element order	Instructs UFT to ignore the order of XML elements when doing the comparison. Select True to ignore the elements or False to notice the element order during the comparison.
Ignore namespaces	Instructs UFT to ignore the XML namespaces of XML elements in the strings. Select True to ignore the namespaces or False to consider the namespaces during the comparison.
Ignore XPaths	A list of all the XPaths that UFT ignores during the comparison. Click the **Add Array** button ✚ to add XPath elements to ignore. You can add as many XPath elements as needed.

Checkpoint Properties

Property	Description
Are Equal	Checks if the values of the **XML String 1** and **XML String 2** properties are equal. Select True if the strings are expected to be equal, or False if they are expected to be different.

XML to String Activity

Relevant for: API testing only

This activity enables you to convert a selected XML string into a regular text string.

The following properties are available for this activity:

Input Properties

Property	Description
Source string	The source XML to convert into a string

The available checkpoint properties depend on the XML loaded in the Checkpoints section.

String to XML Activity

Relevant for : API testing only

This activity enables you to convert a selected text string into XML.

The following properties are available for this activity:

The available input properties depend on the XML loaded in the Input section.

Checkpoint Properties

Property	Description
Source string	The expected XML after the conversion.

Validate XML Activity

Relevant for: API testing only

This activity enables you to validate a selected XML string against a specified XML schema.

The following properties are available for this activity:

Input Properties

Property	Description
XML string	The XML string to validate
Import XSD to test folder	Instructs UFT to import the file (the .xsd file) containing the transformed XML to the folder containing your test.
XSD file	The .xsd file against which the validatation is performed.

Checkpoint Properties

Property	Description
Valid	Indicates whether the XML string specified in the **XML string** property is valid according to the validation performed against the file specified in the **XSD file** property.
Errors	The expected error in the validation.

Activity-Specific Dialog Boxes

Relevant for: API testing only

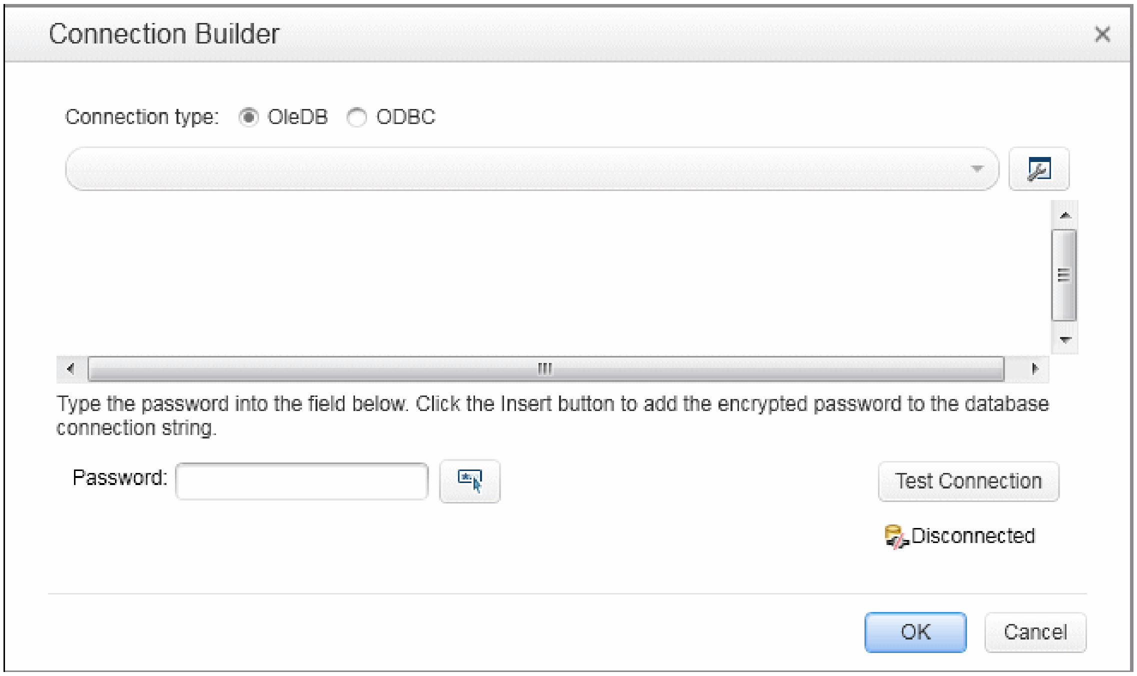
Several activities provide separate dialog boxes to assist you in creating and assigning Input property data.

This section includes:

Connection Builder Dialog Box

Relevant for: API testing only

This dialog box helps you create a database connection expression for the **Database > Open Connection** activity.



Type the password into the field below. Click the Insert button to add the encrypted password to the database connection string.

To access	Do the following: 1. Drag a **Database > Open Connection** activity onto the canvas. 2. Click the **Input/Checkpoints** tab in the Properties pane. 3. Select the Input property—**Connection string**. 4. Click the **Connection Builder** button ... in the right side of the **Value** column.
Relevant tasks	"How to Execute Database Commands or Retrieve Data" on page 1624

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

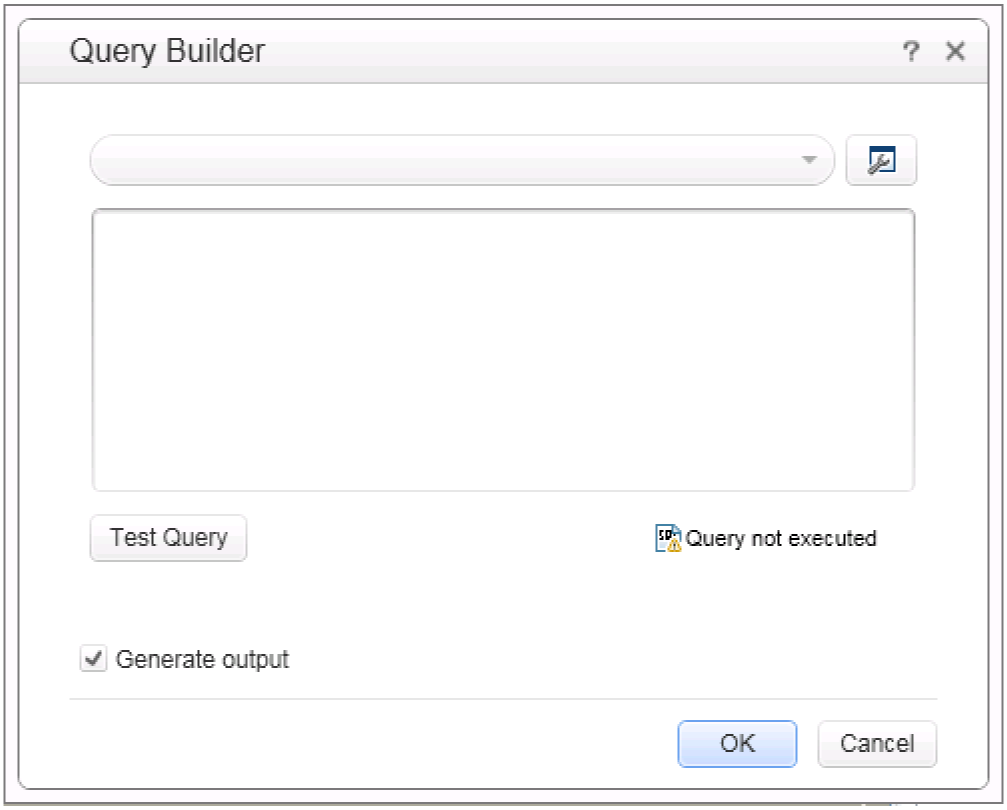
UI Elements	Description
Connection type	The type of database connection: **OleDB** or **ODBC**.
	Edit Connection String. • **For OleDB.** Opens the Microsoft Data Link Properties dialog box to help you create a new string or edit an existing one. • **For ODBC.** Opens the Select Data Source Name dialog box. For details, see "Select Data Source Name Dialog Box" on page 262.

UI Elements	Description
<connection string area>	An editable area for pasting in existing strings, or for editing the connection string selected in the drop down list.
Connection string	A drop down list of previously defined connection strings.
Password	The password required to connect to the database. The password is displayed in encrypted form.
	Insert. Inserts a parameter representing the password at the location of the cursor in the Connection String area.
Test Connection	Checks the database connection using the string shown in the text area. If it succeeds to connect to the database, the status message beneath this button changes from **Disconnected** to **Connected**. After you test a connection, it remains open until after the test is complete or when you close it manually using a **Close Connection** step.
Connected/Disconnected	The current connection status.

Query Builder Dialog Box

Relevant for: API testing only

This dialog box helps you create an SQL statement for the **Query string** property of the **Database > Select Data** activity.



To access	Do the following: 1. Drag a **Database > Select Data** activity onto the canvas. 2. Click the Input/Checkpoints tab in the Properties pane. 3. Select the Input property—**Query string**. 4. Click the **Query Builder** button ... in the right side of the **Value** column.
Relevant tasks	"How to Execute Database Commands or Retrieve Data" on page 1624

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

UI Elements	Description
🔧	**Query Designer.** Opens the Query Designer dialog box.
Query string	A drop-down list of previously defined query strings.
<query string area>	An editable area for pasting in existing strings, or for editing the query string selected in the drop-down list.

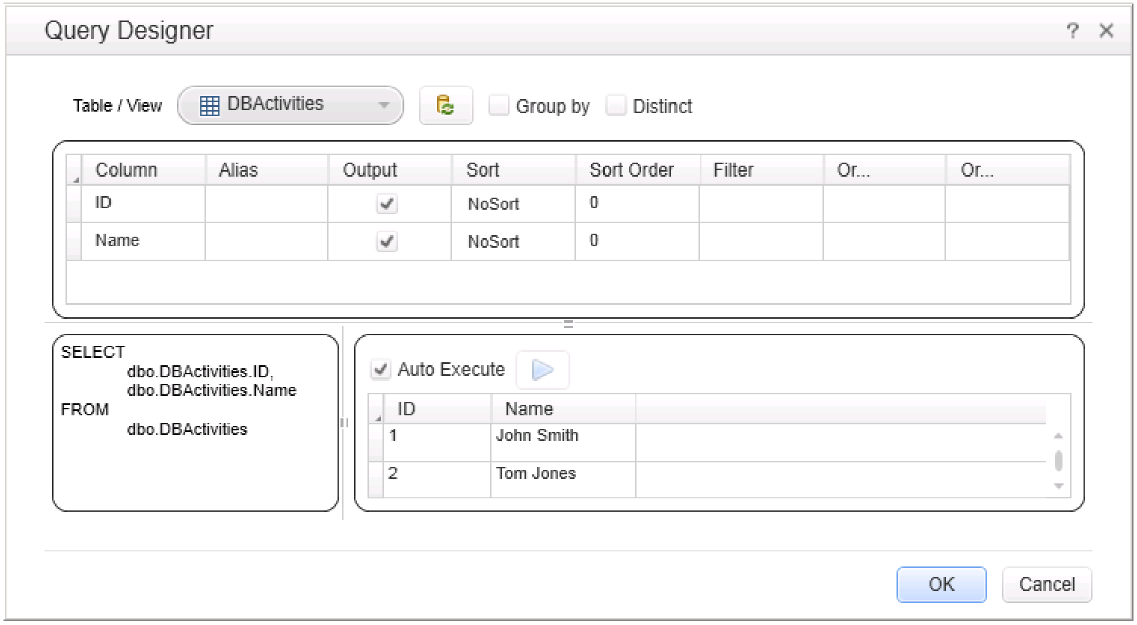
UI Elements	Description
Test Query	Tests the query against the database to which you are connected.
<query status>	The displayed query's status. The possible values are: - **Query not executed** - **Query executed successfully** - **Query completed with errors**
Generate output	Automatically retrieves table information from the database when you click **OK,** and sets the structure of the output in the Properties pane's **Checkpoints** area.

Query Designer Dialog Box

Relevant for: API testing only

This dialog box helps you design an SQL statement for:

- the **Query string** property of the **Database > Select Data** activity.
- creating a new database data source in the Data Pane.



Column	Alias	Output	Sort	Sort Order	Filter	Or...	Or...
ID		✓	NoSort	0			
Name		✓	NoSort	0			

```
SELECT
        dbo.DBActivities.ID,
        dbo.DBActivities.Name
FROM
        dbo.DBActivities
```

ID	Name
1	John Smith
2	Tom Jones

To access	In the Input/Checkpoints tab, for a **Select Data** step: 1. Select the Input property—**Query string**. 2. Click the **Build an SQL query string** button in the right side of the **Value** column to open the "Query Builder Dialog Box". 3. Click the **Query Designer** button . In the Data Pane, when defining a new Database type data source using the "Add New Database Data Source Wizard" on page 258: 1. In the Add New Database Data Source wizard, proceed to the second screen, **Set SQL Statement**. 2. Click the **Query Designer** button to the right of the **SQL statement** box.
Relevant tasks	• "How to Execute Database Commands or Retrieve Data" on page 1624 • "Add a database data source" on page 235

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

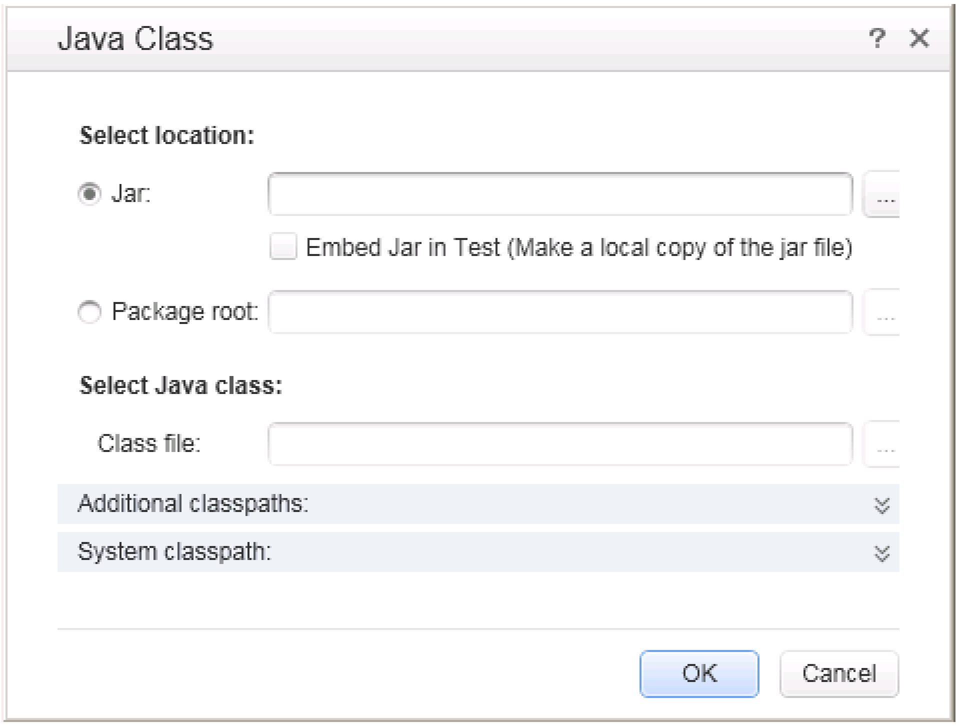
UI Elements	Description
Table/View	A drop-down list of the tables and views in the database. The tables and views are distinguishable by different icons. **Note:** The number of items shown in the list is limited to 1000. You can also enter text in the field to display a list of relevant data sources.
⟳	**Reset Query.** Resets the query to its original state and discards all changes.
Group by	Adds a **GroupBy** column to the grid, allowing you to select a grouping method such as: Sum, Avg, Min, Max, Count, and so forth. It adds GROUP BY to the SQL statement. If you select a **GroupBy** criteria, it also adds it to the SQL expression.
Distinct	Adds a DISTINCT term to the SQL statement, instructing the query to retrieve only distinct record sets.

UI Elements	Description
<table/view columns grid>	A list of the columns in the selected table or view. The grid contains the following columns: • **Column.** The name of the column in the table or view. • **Alias.** An alias name for the column, if applicable. • **Output.** When enabled, it includes the column in the SELECT statement. • **Sort.** Enables you to sort the results in ascending or descending order. • **Sort Order.** When working with multiple sorted columns, this indicates the sorting order beginning with 1. • **Filter.** Enables you to filter the results of the query, by adding a WHERE <column_name> = <filter text> to the SQL statement.
<SQL statement preview pane> (bottom left)	A preview of the SQL statement generated by your selections.
<returned rows pane> (bottom right)	The rows returned by the query. **Tip:** If you enable **Auto Execute**, these rows display the current query results.
Auto Execute	Automatically executes the SQL statement in the **Preview** pane whenever you make a change in the upper panes of the dialog box. **Tip:** If you notice that the query takes a long time to execute, you can disable this option and click the **Execute query** button ▶ to manually run it.
▶	**Execute query.** Executes the query shown in the **Preview** pane. Use this when **Auto Execute** is disabled. **Tip:** If the **Returned rows** pane is grayed out, this indicates that the results are not current.

Java Class Dialog Box

Relevant for: API testing only

This dialog box enables you to configure the settings for a call to a Java class.



To access	Do the following: 1. Add a **Call Java Class** step to the canvas. 2. In the Properties pane, click the Input/Checkpoints tab . 3. Click the **Java Class** button.
Relevant tasks	"How to Create a Call to a Java Class" on page 1627

User interface elements are described below:

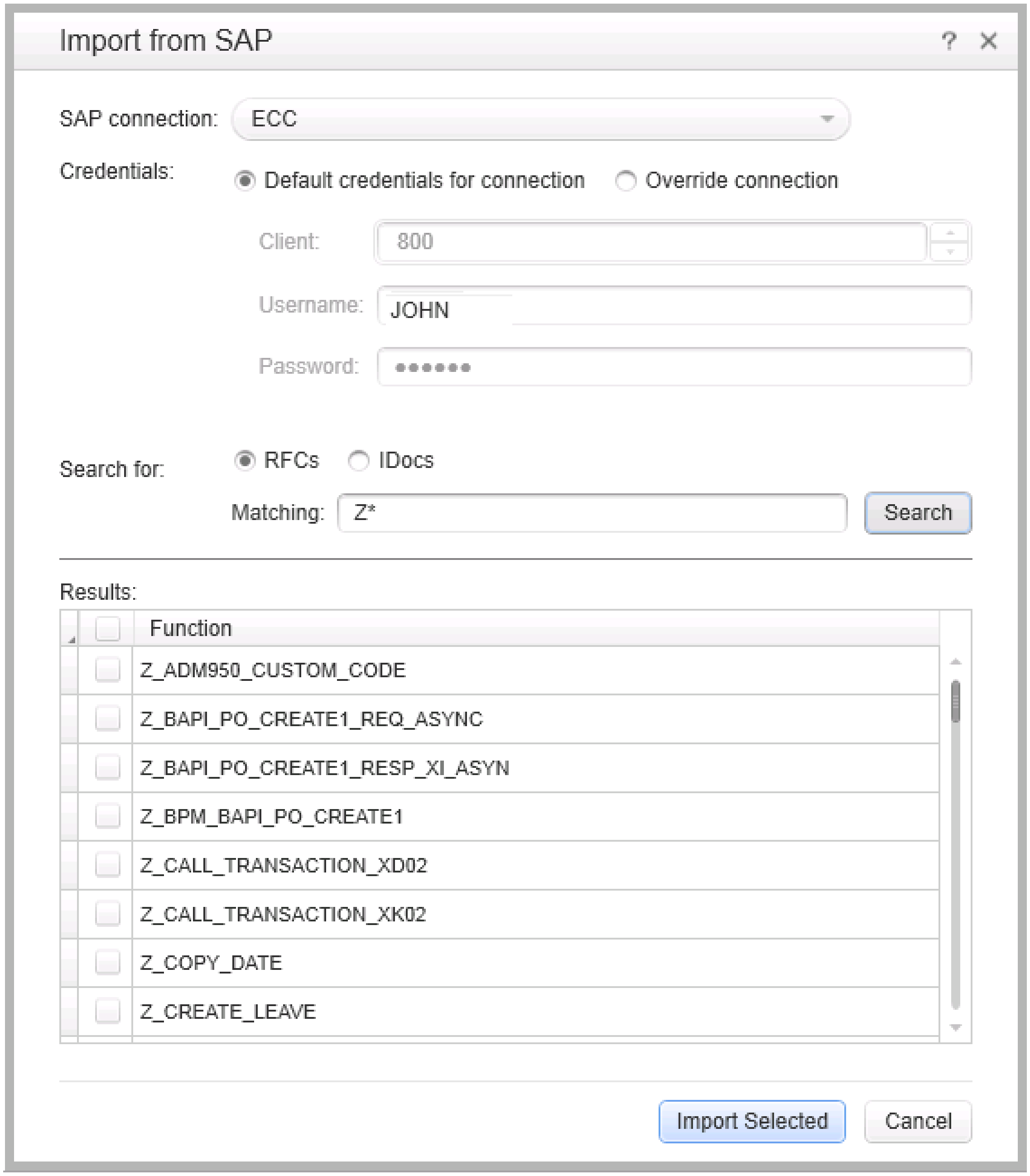
UI Elements	Description
Jar	The Jar (Java Archive) file to use in the Java call.
Embed Jar in Test	Copies the Jar file to the test's **EmbeddedJavaResources** folder. This allows the test to be ported to other machines. (Due to a limitation, you must select this check box before you browse for the class file.)

UI Elements	Description
Package root	The root folder containing the classes to be called by the step.
Class file	The Java class to associate with the **Call Java Class** step. When you click **Browse**, it shows the classes in the specified .jar file. UFT supports the following Java classes: • **Byte** • **Short** • **Int** • **Long** • **Float** • **Double** • **Boolean** • **Date** • **String** **Note:** The **Char** class is not supported.
Additional classpaths	Additional classpaths to associate with the **Call Java Class** step. • **Jar.** Opens the **Select Jar File** dialog box. • **Folder.** Opens the **Browse for Folder** dialog box to select a folder containing the **.class** file. • **Add.** Adds the contents of the **Additional Classpath** box to the additional classpath list. • **Remove.** Removes the selected entry from the list.
System classpath	The paths you defined in the Test Settings for system classpaths. To modify this setting, click on the **Start** or **End** step and open the Properties pane's **General** view. Modify the Test Setting's **JVM > Classpath** node.

Import from SAP/Update Dialog Box

Relevant for: API testing only

This dialog box enables you to import an SAP IDoc or RFC into the Toolbox pane or update an existing one from a new location.



Import from SAP

SAP connection: ECC

Credentials: ● Default credentials for connection ○ Override connection

Client: 800

Username: JOHN

Password: ●●●●●●

Search for: ● RFCs ○ IDocs

Matching: Z*

Results:

□	Function
□	Z_ADM950_CUSTOM_CODE
□	Z_BAPI_PO_CREATE1_REQ_ASYNC
□	Z_BAPI_PO_CREATE1_RESP_XI_ASYN
□	Z_BPM_BAPI_PO_CREATE1
□	Z_CALL_TRANSACTION_XD02
□	Z_CALL_TRANSACTION_XK02
□	Z_COPY_DATE
□	Z_CREATE_LEAVE

To access	**To import a new RFC or IDoc:** 1. Define an SAP Connection. For details, see "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 568. 2. Select **Tools > Import from SAP**. **To update an RFC or IDoc from a different location:** 1. Select the RFC or IDoc in the Toolbox pane. 2. Select **Update from** from the right-click menu.
Relevant tasks	"How to Create an SAP API Test Step" on page 1638

Some user interface elements are only available when you select a specific node or if you import a new RFC or IDoc. The user elements are:

UI Elements	Description
SAP Connection	A drop down containing the names of the SAP connections defined in the "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 568 (read-only).
Credentials	The credentials to use for accessing the server: • **Default credentials for connection.** Use the client and credentials defined for the connection. • **Override connection.** Manually provide client and credential information for this import.
Search for	The type of document to search for: **IDOC** or **RFC**.
Matching	The matching condition. Use an asterisk, *, as a wildcard.
Search	Begins the search based on the matching conditions.
Results	The IDOCs or RFCs that matched the condition.
Import Selected	Imports the RFCs or IDocs indicated with a check mark into the **Local Activities > SAP** section of the toolbox.
Update	Updates the selected RFC or IDoc.

Troubleshooting and Limitations - Activities

Relevant for: API testing only

This section describes troubleshooting and limitations for working with activities.

System Activities

- **End Program.** You cannot specify **Window Title** as an input method for a windowless process, even if you are using a wildcard expression.
- **End Program.** If you are running on a 64-bit machine, the **End Program** activity will not be able to terminate 64-bit applications. However, it can terminate other 32-bit applications.

Java Activities

- **Call Java Class.** Supports only Java primitive types.
- **Call Java Class.** Once you select a Java file for the call, the **Java Class** button is disabled. As a result, you cannot replace or update the Java file.

  Workaround: Remove the step containing the **Call Java Class** step and add a new one using the new Java file.
- **Call Java Class.** Java code loaded by the JVM (Java Virtual Machine) cannot be modified or updated when the JVM is running.

Network Activities

- **HTTP Request/Receiver.** Nested transactions are not support ed.

  Workaround: Add a new loop activity within an existing transaction. Add the new transactions steps to the newly created loop. Make sure to set the loop iteration to 1.
- **HTTP Request/SOAP Request**. XML file that use XSL, are not supported.
- **SOAP Request.** Switching between **Text** and **Grid** views, may cause the grid to display an element added below the Body's **Any** node, under the Header's **Any** node.

  Workaround: Open the **Text** view to view the correct XML..
- **SOAP Request.** By default, UFT validates the request received in a SOAP Request step using the SOAP 1.1 schema. If you expect your response to use the SOAP 1.2 schema, the validation will fail.

  Workaround: Import the SOAP 1.2 schema for your SOAP Request step's response body. This schema is available with the UFT installation at <UFT installation folder>\Addins\ServiceTest\WSImportTechnology\envelop1.2.xsd.

Database Activities

- Nested transactions are not supported in database transaction activities.

  Workaround: Add a new loop activity within an existing transaction. Add the new transactions steps to the newly created loop. Make sure to set the loop iteration to 1.

- Databases which are supported by ODBC, but not by OLEDB, cannot be accessed by UFT.

FTP Activities

- **FTP:** When working with FTP activities that specify paths, you need to enter the full path.

- **FTP Download:** When downloading from Linux servers, you cannot download an empty folder.

Load Testing Activities

- Related data mapping in a load-enabled test is not supported for the LoadRunner parameter advance policy of **Each Occurrence**.

- Tests created as Business Process Testing (BPT) components, cannot be used in load testing.

- If your tests use IBM's MQ client, make sure to install the MQ client on all machines running these tests.

- You cannot run tests containing actions or calls to other tests on a remote load generator. This limitation does not apply when running the test on a local load generator.

- The data assignment method, **Use a unique value for each Virtual User when load testing** is not supported in all environments.

IBM Websphere MQ Activities

- **Get Message from MQ Queue.** The MQGMO_MARK_SKIP_BACKOUT option is not supported.

- **Put Message to MQ Queue, Publish Message to MQ Topic.** The message body should not exceed 64K bytes. If it exceeds this size, the execution issues a MQRC_CONVERTED_ STRING_TOO_BIG status. This is a limitation of IBM MQ.

- **MQ Steps.** Automatic linking of IBM Websphere MQ steps to the most recent **Connect to MQ Queue Manager** connection, is supported only when the steps are on the same level in the container, or if the connection step is in a parent container. If the connection step is in a leaf container and the step using the connection is in a parent container or in another leaf, UFT does not create an automatic link.

  Workaround: Manually link to a MQManager property using the Select Link Source dialog box.

XPath Activity Checkpoints

XPath aggregate functions are not supported.

Chapter 56: Local Activities

Relevant for: API testing only

This chapter includes:

Concepts

Local Activity Overview

Relevant for: API testing only

In addition to the **Standard** activities, you can create or import other services and activities. These activities, displayed under the Toolbox pane's **Local Activities** node, are Web Service, REST Service, Web Application Services, .NET assembly operations, SAP IDocs/RFCs whose contract/document or DLL you import.

You generate your own **Local** activities by defining or importing contract or document files. UFT supports a number of different types of Local activities: Web Services, REST Services, Web Application Services, NET Assemblies, and SAP. In addition, UFT also enables you to create activities based on a network traffic capture. After you import or define at least one entry per type, the Toolbox pane displays its type under the Local activities node.

This section includes:

- "Web Service Activities " below
- "REST Service Activities" on the next page
- "Web-Application Services" on the next page
- "Network Capture Activities" on page 1736
- ".NET Assembly Activities" on page 1736
- "SAP Activities" on page 1737

Web Service Activities

You typically begin creating a test by importing a WSDL file. This file provides a structure for the test, since it describes the service in terms of its elements, argument values, and properties. For details on importing a WSDL, see "How to Import a WSDL-Based Web Service" on page 1744.

If your service changed during development, you can use the Update tool for synchronizing the service. For details, "Updating Services and Assemblies" on page 1853.

The WSDL import supports both Document/Literal and RPC type Web services.

In **Document/Literal** Web services, the client sends standard XML documents to the server. The server application is responsible for mapping the server objects (properties, method calls, and so forth) and the values in XML documents. The data is serialized according to a given schema, so it can be validated against the schema. For Document/Literal type Web services, the Properties pane displays the input and output properties in a grid, allowing you to assign values for each property independently.

In **RPC** type Web services, the WSDL file and SOAP body contain the complete operation name, its input and output properties, and their values. There is no schema for this type of service and it is not supported by the WS-I conformance standard. The Properties pane does not display the input and output properties for RPC type services.

If your service document is unique and cannot be imported in the normal way, you can use the SOAP Request activity to send a manual SOAP request to the server.

The WS-I validation tool lets you validate imported services by checking their conformance with the WS-I standard. For details, see "Validate the WSDL file - optional" on page 1745.

After you import a Web service, the tree hierarchy displays its components in several levels:

- **Level 1.** Service name
- **Level 2.** Port number
- **Level 3.** Operation name

For details about importing a Web service, see "How to Import a WSDL-Based Web Service" on page 1744.

REST Service Activities

You can define REST services, their resources, and methods. You define the metadata manually by modeling the service and defining your own resources and methods. The services, resources, and methods are then stored as a prototype activity within your test.

After you define a REST service and its resources and methods, UFT creates a hierarchy of **Service**, **Resource**, and **Method**, based on your definition of the REST service.

UFT provides a designer window for creating and configuring REST service method activities. You drag the REST methods onto the canvas, just as you would with other Toolbox pane activities.

After creating REST methods, you can reuse them in multiple steps. For details, see "How to a Create a REST Service" on page 1746.

UFT also enables you to update REST service definitions and resolve service definition conflicts through the "Resolve REST Method Steps Wizard" on page 1871.

In addition, you can also define properties and parameters for your REST service at all levels of the hierarchy. You can then pass these properties or parameters from the service and resource levels to the resource and method levels. For details, see "Passing REST Service Properties" on page 1738.

Web-Application Services

Web Application services provide a description of a HTTP-based Web application in an XML format. This description is contained in a Web Application Description Language (WADL) file. The WADL file describes the resources provided by a service and the methods used to access the service.

Like a Web Service, you import a Web application service into UFT. The resources and methods are then displayed in a hierarchy like a REST service, in a Service, Resource, and Method hierarchy.

The URL for a Web application is defined in the XML of the WADL file. You can, however, define other HTTP properties and add input and output parameters for an activity's methods.

Like REST services, you can define parameters and their values at all levels of the Web Application hierarchy. You can then pass these parameter values to lower levels of the hierarchy.

The imported Web Application service methods serve as a prototype for test steps. You can modify the parameter values of a method after dragging a method to the canvas.

Network Capture Activities

Network capture activities enable you to create test steps by recording network traffic. Importing a network capture file provides you with another way to create test steps measuring the network activity of your application or Web service. Instead of using the standard Network activities to design steps for your application's network processes, you can perform a network capture and use the captured information as a basis for your test.

Using a network capture program, you capture the network traffic for your application or Web service. The network capture program then saves the file containing the network traffic. You then import this file into UFT.

UFT takes the TCP network stream and creates test steps based on the request and response information for each TCP stream in the network traffic capture.

Based on the request and response information, UFT creates a test activity differently:

- If the TCP stream request and response is compatible with or matches an already existing Web service, UFT creates a **Web Service** step.
- If the TCP stream request has a SOAP request structure, UFT creates a **SOAP Request** step.
- If the TCP stream is not similar to an existing Web service method or a SOAP request transaction, UFT creates a **HTTP Request** step.

These activities are not stored in the Toolbox pane. If you need to reuse the steps in your test, you can reimport the network capture file or cut and paste the existing steps into your test.

For task details, see "How to Import a Network Capture File" on page 1758.

.NET Assembly Activities

The .NET importer lets you create activities for testing APIs in the form of .NET assemblies. You can interface with the types defined in the assembly.

You begin by importing the .NET assembly into your test. The Toolbox pane displays the assembly as an activity. You drag the .NET activity on to the canvas as you would with any other activity.

You can define input and output properties and link them to other steps within your script. For example, you can design a test that retrieves an object through the .NET assembly. This object can be an input property for a Web service step.

When you import a .NET assembly, it saves a local copy of the assembly with the test. This makes the test portable and allows you to copy it to another machine. If the assembly calls other assemblies, the test may not run until you copy the additional assemblies to the new machine.

For task details, see "How to Import and Create a .NET Assembly API Test Step" on page 1760.

For user interface details, see "Import .NET Assembly Dialog Box" on page 1774.

SAP Activities

The SAP importer lets you create SAP activities in the Toolbox pane, by importing SAP Intermediate Documents (IDoc) and Remote Function Calls (RFC).

These activities can be useful for testing the SAP server response in several common scenarios:

- Sending an IDoc to an SAP server, and confirming that the IDoc was sent
- Checking an IDoc's status on the SAP server
- Calling an RFC in SAP and ensure that it returned the expected results

These activities can be also be useful when upgrading a system to verify that the integration patterns are still functional, such as Aggregator, Enricher, Router, Translation, Bridge, Splitter, and so forth.

First you define one or more SAP Connections in the SAP Connections pane in the Options dialog box (**Tools > Options > API Testing** tab **> SAP Connections** node). For details, see "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 568.

Once the connection is established, you can search for an IDoc or RFC and import them as activities into the Toolbox.

Once they are in the Toolbox, you can add them to a test or action and assign property values as with all other steps.

For task details, see "How to Create an SAP API Test Step" on page 1638.

For user interface details, see "Import from SAP/Update Dialog Box" on page 1729.

Activity Sharing

Relevant for: API testing only

The activity sharing feature enables you to save locally stored services to a repository, so that they will be available for other tests.

You can specify a repository on the file system or in ALM. The next time you create a test, you can access the service's activities from the repository instead of reimporting or recreating them.

If the resource (WSDL or REST service) becomes unavailable, the canvas displays alerts on the steps that use the resource. If you run the test when its resource is not available from its original source, it uses a copy of the test stored in its cache. By clicking the alert, you can reload the steps when the resource becomes available again.

The Toolbox pane detects version updates for activities stored in a repository. When it detects a discrepancy between the step and the Toolbox pane activity, it displays an alert for the step. By clicking the alert, you can automatically update the resource from its source.

Negative Testing of Web Services

Relevant for: API testing only

When performing a functional test for your Web Service, you should approach the testing in a variety of ways. The most common type of testing is called **Positive Testing**—checking that the service does what it was designed to do.

In addition, you should perform **Negative Testing**, to confirm that the application did not perform a task that it was not designed to perform. In those cases, you need to verify that the application issued an appropriate error—a SOAP Fault.

To illustrate this, consider a form accepting input data—you apply positive testing to check that your Web Service has properly accepted the name and other input data. You apply negative testing to make sure that the application detects an invalid character, for example a letter character in a telephone number.

When your service sends requests to the server, the server responds in one of the following ways:

- **SOAP Result.** A SOAP response to the request.
- **SOAP Fault.** A response indicating that the SOAP request was invalid. Negative Testing applies only to SOAP faults.
- **HTTP Error.** An HTTP error, such as Page Not Found, unrelated to Web Services.

UFT can check for a standard SOAP result or a SOAP fault response. For example, if your Web Service attempts to access a Web page that cannot be found, it will issue a 404 HTTP error. Using negative testing you indicate that you expect a SOAP fault. In this case, the test run will fail if the service accesses a *valid* Web page.

The Properties pane lets you provide values for the SOAP fault header and body. You can enter **faultcode**, **faultstring**, and **faultactor** values as well as custom properties using **Any** type parameters. Using the Checkpoint mechanism, you can validate these values and view the results in the Run Results Viewer.

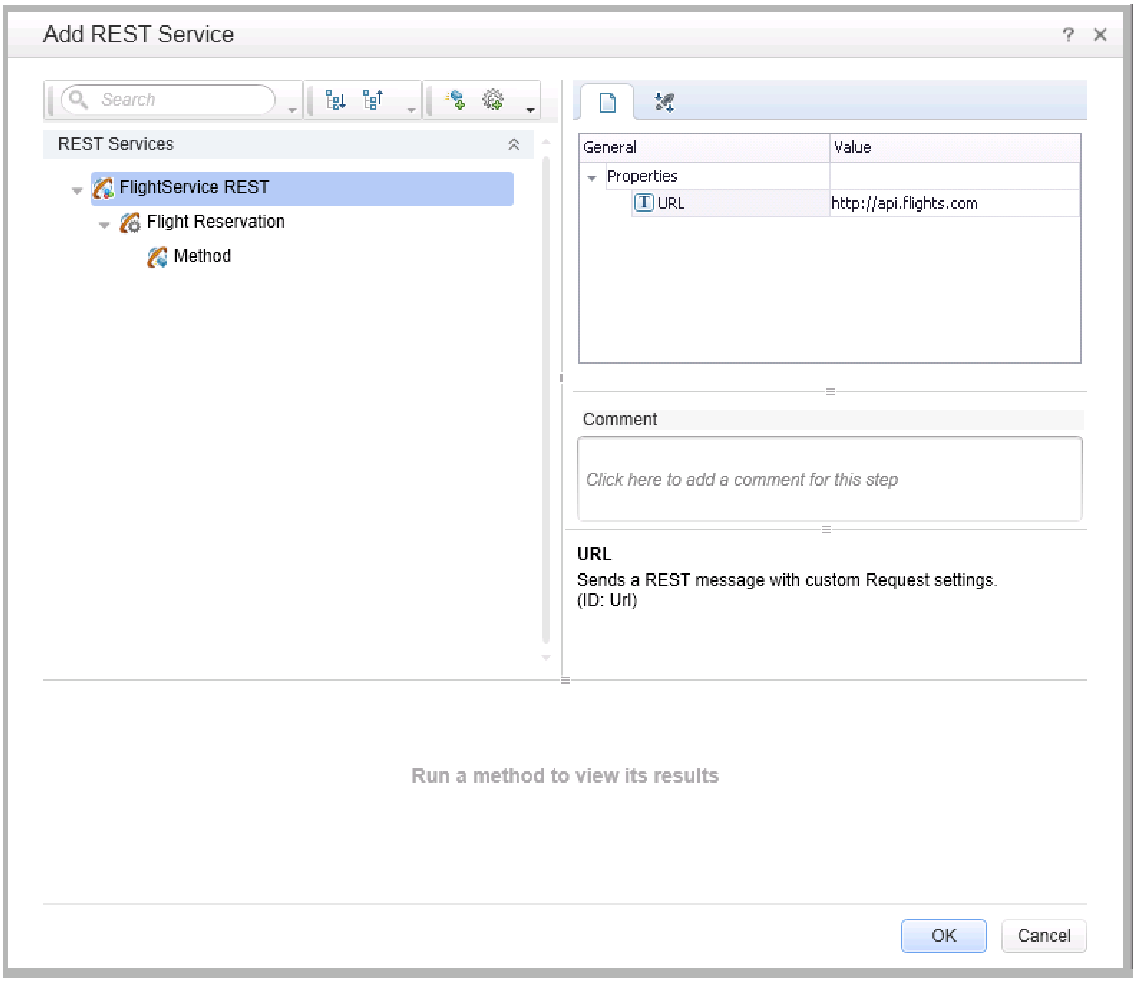
For details on setting up the SOAP fault values, see "SOAP Fault Tab (Properties Pane - API Testing)" on page 428.

Passing REST Service Properties

Relevant for: API testing only

When creating or editing a REST service, you may want to define the value of a URL property or a custom input or output property at the service or resource level in order to make this property or parameter available for all resources and methods included in the service or resource. For example, if your REST service resources and methods are all referencing the same URL prefix during run time, you can define the value of URL property at the service level of the REST service and pass the URL property value to all its resources and methods.

You can also define custom input and output property names and values at all levels of the hierarchy and pass these input and output properties and their values through the REST service hierarchy. For details on creating custom input and output properties, see "Define custom properties - optional" on page 1748.



General	Value
Properties	
URL	http://api.flights.com

Click here to add a comment for this step

URL
Sends a REST message with custom Request settings.
(ID: Url)

After a URL property value is defined at a higher level, you can define other (relative) additions to the URL property value for any of the resources and methods below it. These adjusted relative URL values are then concatenated to the URL property value that was received from higher levels of the hierarchy and the adjusted values are passed to all levels below it. For example, if you add a URL property value at the service level, you can append relative URL paths for a selected resource or method. The URL property value passed from the service level is then concatenated with the relative URL property value added at the resource or method levels.

> **Note:** You cannot modify URL property values passed down from a higher level of the REST service hierarchy. You can only add to them with a relative URL value.

After you add additional relative URL property values at lower levels of the hierarchy, such as at the resource or method level, you must assure that the full URL is a proper URL. For example, if you define the URL property value at the resource level with http://, the relative URL property value appended at the method level should not also use a http:// prefix.

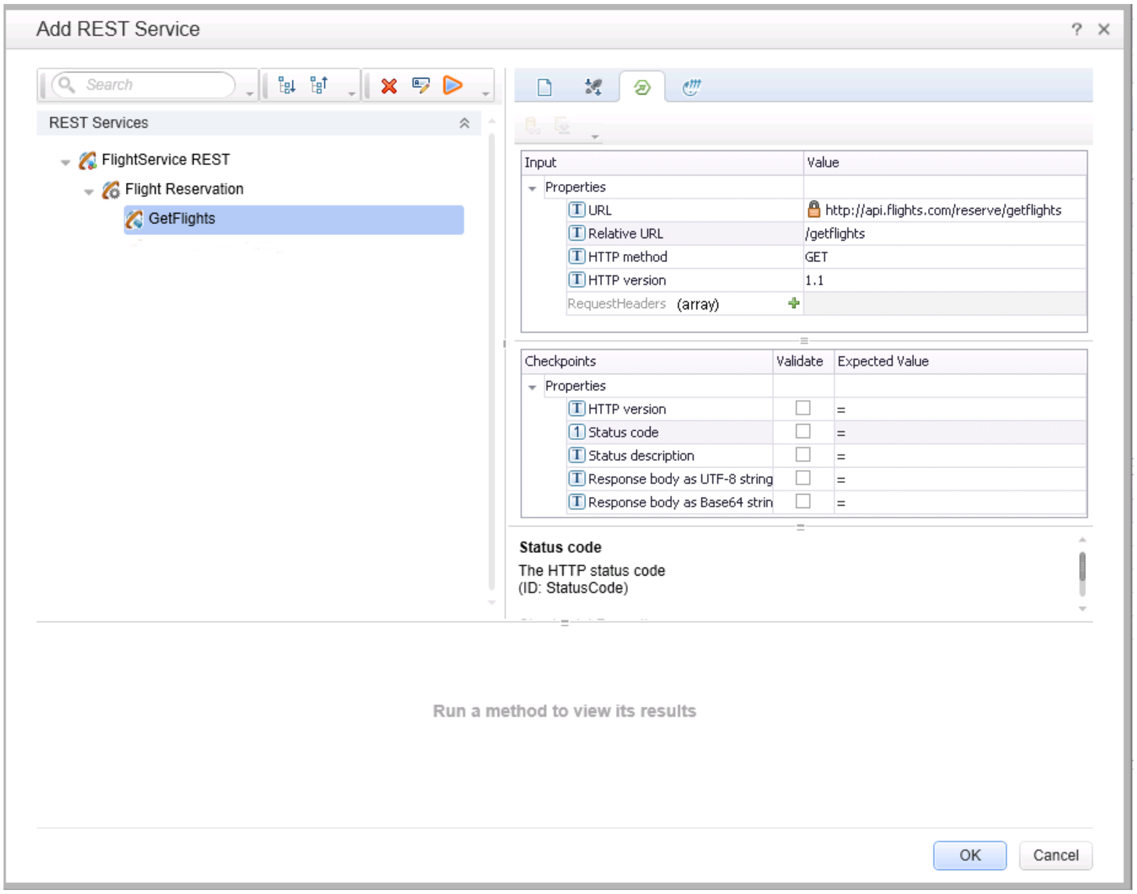
Example

You define a URL property value for the REST service at the service (top) level: http://flights.api.com. This value is then passed to any resources and methods within this REST service.

You also create a resource for this service, called **Flight Reservation**. You define the relative URL value as /reserve. This relative URL property value is then concatenated with the URL property value passed from the service level to create a URL for the resource: http://flights.api.com/reserve. This URL property value can also pass to any methods created for the resource.

You then create a method for the Flight Reservation resource, called **GetFlights** and define the relative URL property value for this method as /getflights. This value is then further concatenated with the URL passed from the resource to make a complete URL for the method: http://flights.api.com/reserve/getflights.

The example below shows the URL property value passed from the service level (http://api.flights.com) and the relative URL property value defined at the resource level (/reserve), with the relative URL property value for the method (/getflights) also defined. These URL parts are then concatenated in the **URL** property field to make the complete URL for the method.



Input	Value
Properties	
URL	http://api.flights.com/reserve/getflights
Relative URL	/getflights
HTTP method	GET
HTTP version	1.1
RequestHeaders (array)	

Checkpoints	Validate	Expected Value
Properties		
HTTP version	□	=
Status code	□	=
Status description	□	=
Response body as UTF-8 string	□	=
Response body as Base64 strin	□	=

Status code
The HTTP status code
(ID: StatusCode)

These properties and custom input or output properties then serve as a prototype template for the REST service, and you can edit the URL property values or custom input and output property values after dragging the method activities in the canvas.

> **Note:** The relative URL is not displayed on the REST method when it is included in a test in the canvas. Only the full, concatenated URL for the method displayed in the Add REST Service dialog box is displayed in the Properties pane for the selected method.

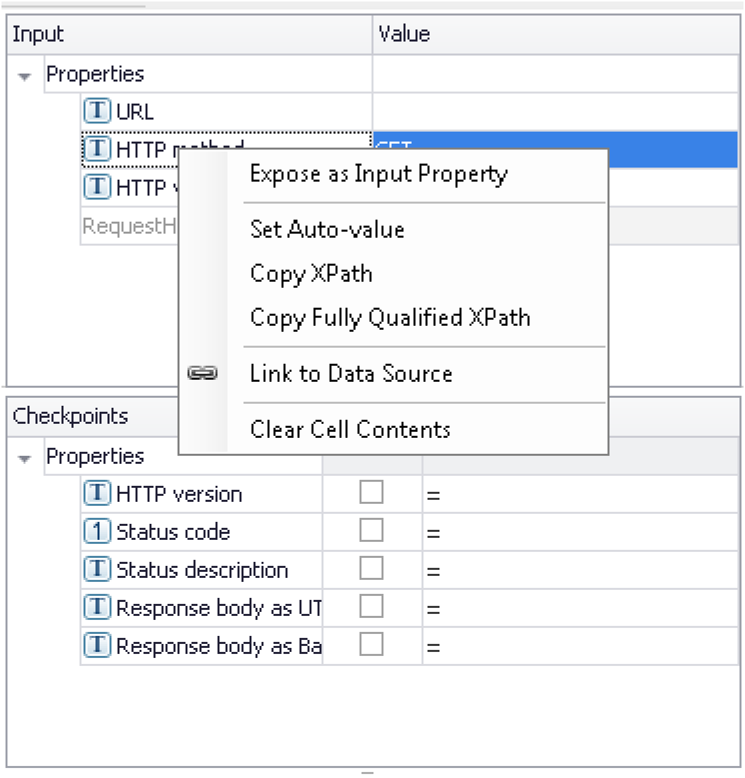
Exposing REST Service Properties

Relevant for: API testing only

When working with REST service methods saved in API tests created in UFT 11.51 or earlier or Service Test 11.51 or earlier, the Properties pane enables you to expose input and output HTTP properties. Exposing HTTP properties means that you make them available at the REST method wrapper level instead of just the inner HTTP level. You can expose properties that are available from the **General**, **Input/ Checkpoint**, **HTTP**, and **Multipart** views.

> **Note:** You can only expose properties if you are working withAPI tests created inUFT 11.51 or earlier or Service Test 11.51 or earlier.

The following example shows the shortcut menu item, **Expose as an input property**. This option prompts you to provide a name for the property as it should appear in the REST wrapper level.



The property, **HTTP method**, will be available in the REST method wrapper. To see the exposed property, select the REST method wrapper in the canvas —not the inner HTTP Request.

> **Note:**
>
> - When exposing a property with incoming links, the links are redirected to the newly created property.

- When exposing a complex property, the new property will be created as a **String** type.

For user interface details, see "New Exposed Property Dialog Box" on page 1850.

Tasks

How to Import a WSDL-Based Web Service

Relevant for: API testing only

This task describes how to import a WSDL file into your test.

This task includes the following steps:

- "Prerequisites" below
- "Open the Import WSDL dialog box" below
- "Select a source " below
- "Set connection settings for URL/UDDI imports - optional" on the next page
- "Mark the WSDL as a server response - optional" on the next page
- "Complete the import" on the next page
- "Validate the WSDL file - optional" on the next page
- "Configure SOAP Fault information - optional" on page 1746

1. Prerequisites

If you intend to import a WSDL stored on ALM, the ALM project to which you are connecting must have the Service Test Management extension enabled. For details, see the "ALM Connection Dialog Box" on page 737.

2. Open the Import WSDL dialog box

- To import a WSDL from the file system or ALM, click Import WSDL ▾ **> Import WSDL from File or ALM Application Component**. For details, see the "Select WSDL Dialog Box" on page 1764.

- To import a WSDL from a Web site or UDDI repository, click Import WSDL ▾ **> Import WSDL from URL or UDDI**.

  - For a URL, select **Import from: URL**
  - For a UDDI, select **Import from: UDDI**

  For details, see the "Import/Update WSDL from URL or UDDI Dialog Box" on page 1766.

3. Select a source

- For **File System** or **ALM Application Components** imports, click the appropriate button in the left pane. Navigate to the WSDL in the file system or the ALM repository.

- For a **URL** import, click the **Browse** button adjacent to the **Address** field. This opens a browser window. Navigate to the WSDL and close the browser. This automatically places the URL in the **Address** field.

- For a **UDDI** import, click the **Browse** button adjacent to the **Address** field. This opens the Select Service from UDDI dialog box. Specify a UDDI address and select the service to import. For details, see the "Select Service from UDDI Dialog Box" on page 1767.

4. **Set connection settings for URL/UDDI imports - optional**

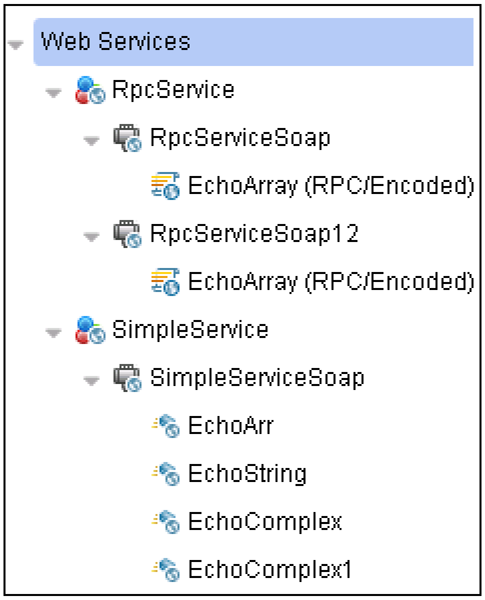
   For URL or UDDI imports, if your WSDL must be accessed through a secure server or proxy machine, click **Advanced Settings** to expand the dialog box. Enter the authentication information or the proxy server details.

5. **Mark the WSDL as a server response - optional**

   You can mark the imported WSDL as a server response method. This is useful when working with asynchronous Web services. Click **Advanced Settings** to expand the dialog box and select the **Import as server** option. For details about working with asynchronous services, see "Asynchronous Services" on page 1928.

6. **Complete the import**

   Click **OK**. If the import succeeds, the **Toolbox** displays the service, its ports, and operations under the **Web Services** node. The Toolbox pane differentiates between Document/Literal and RPC encoded services with different icons.



- Web Services
  - RpcService
    - RpcServiceSoap
      - EchoArray (RPC/Encoded)
    - RpcServiceSoap12
      - EchoArray (RPC/Encoded)
  - SimpleService
    - SimpleServiceSoap
      - EchoArr
      - EchoString
      - EchoComplex
      - EchoComplex1

7. **Validate the WSDL file - optional**

To check the WSDL's compliance with the WS-I standard, right-click the service and select **Validate WS-I Compliance** from the shortcut menu. The Output window shows the WS-I validation results. These validations apply only to Document/Literal type services, but not RPC.

8. Configure SOAP Fault information - optional

To apply negative testing to a Web service:

a. Open the **SOAP Fault** tab .

b. Select **Fault is expected**.

c. Provide SOAP Fault checkpoint values for the negative testing:

- To work In the **XML** layout: Expand the SOAP nodes and define **Any** elements for the SOAP Header and Body. If relevant, provide values for **faultcode**, **faultstring**, or **faultactor**.

- To work with **XPath** expressions: Click the **XPath** tab and use the Add button to add new XPath entries. Copy the XPATH entry into the cell.

For details, see the "SOAP Fault Tab (Properties Pane - API Testing)" on page 428.

How to a Create a REST Service

Relevant for: API testing only

This task describes how to set up a REST service with its resources and methods. You define the request body and properties in order to create a prototype method that could be reused by multiple test steps.

This task includes the following steps:

- "Prerequisite " on the next page
- "Open the REST service designer" on the next page
- "Name the REST service" on the next page
- "Add a resource" on the next page
- "Add a method" on the next page
- "Set the General properties" on the next page
- "Set the URL property values" on page 1748
- "Define the method's HTTP properties" on page 1748

- "Define custom properties - optional" on the next page
- "Enter the request body directly - optional" on the next page
- "Link the body request to a data source - optional" on page 1749
- "Test the method" on page 1750
- "Add the method to the toolbox" on page 1750
- "Use the REST activity" on page 1751
- "Expose input and output properties - optional (for API tests created in previous versions)" on page 1751
- "Edit the prototype service" on page 1751
- "Resolve conflicts between the prototype and the test step - optional" on page 1752
- "Share the service - optional" on page 1752

1. **Prerequisite**

   Study the structure of the REST Service body and determine which resources and methods you need to define.

2. **Open the REST service designer**

   Click the **Add REST Service** toolbar button . The Add REST Service dialog box opens. For details, see the "Add/Edit REST Service Dialog Box" on page 1772.

3. **Name the REST service**

   Click on the **New Service** node and provide a meaningful name for the service in the left pane.

4. **Add a resource**

   Click the **Add Resource** toolbar button to add a resource to the REST service. Click on the **REST Resource** node and provide a meaningful name for the resource.

5. **Add a method**

   Click the **Add Method** toolbar button to add a method. Click on the **Method** node and provide a meaningful name for the method.

6. **Set the General properties**

   a. In the right pane's **Properties** list, open the **General** tab.

   b. Enter values for the properties. For details, see the "General Tab (Properties Pane - API

Testing)" on page 413.

7. **Set the URL property values**

   a. In the REST Service editor, select a REST service, resource, or method.

   b. In the General tab, enter the URL property value:

      - If you are entering a URL property value for a REST service, enter the URL prefix in the **URL** property row, beginning with a http://

      - If you are entering a URL property value for a REST resource or method, enter the URL property value in the Relative URL property row.

   > **Note:** If you enter a URL property value for the service or resource of your REST service, the URL property values are passed to all resources or methods included in the service or resource. For details, see "Passing REST Service Properties" on page 1738

8. **Define the method's HTTP properties**

   a. Select a REST service method.

   b. In the REST service editor's right pane,click the **HTTP Input/Checkpoints** tab.

   c. For each method, modify the **HTTP method** and **HTTP version**.

   d. Use the plus sign ✚ in the **RequestHeaders** parent node to add name and value pairs for the request header array.

      For details, see the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

9. **Define custom properties - optional**

   For methods that require input, such as PUT or POST, you can add custom input properties. For methods that provide an output such as GET, you add the required output properties. To create these properties:

   a. In the right pane's **Properties** list, click the **Custom Input/Checkpoints** tab .

   b. Expand the plus button ✚ and select **Add Input/Output Property**.

   c. Provide a name, data type, and description (optional) for each property.

   d. Enter the property values in the **Value** column.

10. **Enter the request body directly - optional**

This step describes how to enter the request body directly. (For details on linking to a data source, see the next step.)

a. Return to the design document for the REST service and copy the Request body onto the clipboard.

b. In the right pane of the Add REST Service window, open the **HTTP** tab. Make sure the **Body** type in the drop down is set to **Text**, and click within the **Body** section. Press **CTRL+V** to paste the contents of the request into the pane. Make sure that there are no extra spaces before or after the body text.

c. Modify the element values within the XML body as required.

11. Link the body request to a data source - optional

This step describes how to enter the Request body through a data source. (For details on entering the Request body directly, see the previous step.)

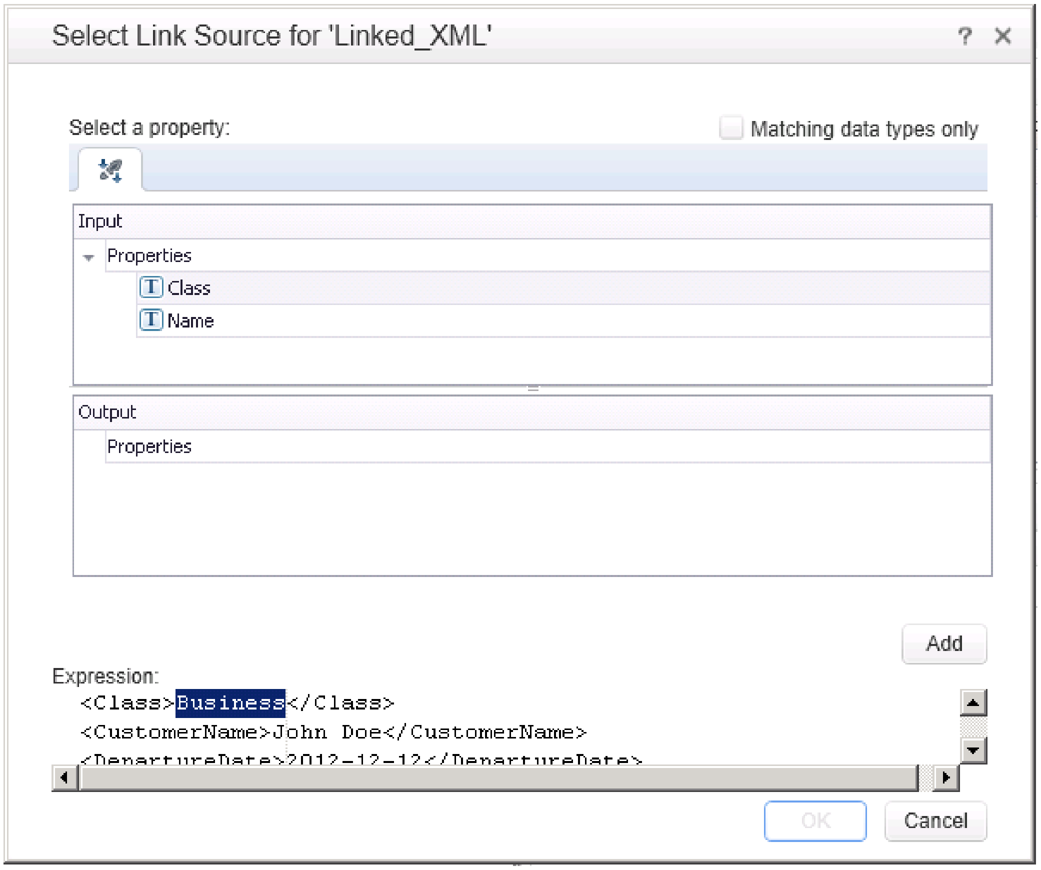
a. Return to the design document for the REST service and copy the request body to the clipboard.

b. In the right pane of the Add REST Service dialog box, open the **HTTP** tab . Make sure the **Body** type in the drop down is set to **Text**, and click within the **Body** section.

c. Click the **Link Body** button 🔗 to the right of the pane, to open the Select Link Source dialog box. For details, see "Select Link Source Dialog Box (API Testing)" on page 1840.

d. In the Select Link Source dialog box, click **Custom Expression** to expand the dialog box**.** Paste the clipboard contents, the Request body, into the **Expression** area.

e. In the **Expression** area, highlight the value of the element you want to link. In the upper pane, double-click on the custom property you defined earlier in the properties list. The property is now linked to a value.



```
<Class>Business</Class>
<CustomerName>John Doe</CustomerName>
<DepartureDate>2012-12-12</DepartureDate>
```

f. The modified expression appears in the **Expression** area. In the following example the custom Class property is linked to Class element in the REST document.

```
<Class>{Step.InputProperties.RestMethod.Class}</Class>
```

g. Click **OK**. UFT places the data in the **Body** area.

12. Test the method

Click the Run Method button on the toolbar to test the method with its property values. Note the results in the lower section of the window. If something needs to be modified, do so at this point.

13. Add the method to the toolbox

Click **OK** in the Design REST Service dialog box. UFT adds the REST service along with its resources and methods to the Toolbox pane, under the **Local Activities** category.

14. ## Use the REST activity

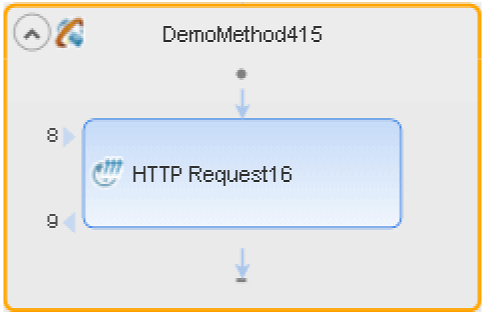
    You have now created a prototype for a REST method, complete with input parameters and HTTP information. You can now drag the activity, whose primary properties are already defined, onto the canvas. A wizard helps you resolve any conflicts between the original prototype and a test step. For details, see "How to Resolve Conflicts in a REST Service Test Step" on page 1860.

15. ## Expose input and output properties - optional (for API tests created in previous versions)

    You can forward built-in HTTP properties to the REST method wrapper. This is useful for making specific HTTP properties available at the wrapper level.

    > **Note:** If you created a REST method in UFT 11.52 or Service Test 11.52 or later, all REST properties are incorporated into the REST activity step itself without a wrapper.

    a. Double-click a REST method from the Toolbox pane to add it to the canvas. Expand the method to show the **HTTP Request** frame.

    b. Click in the **HTTP Request** frame. Select a property in the Properties pane and select **Expose as Input Property** or **Expose as Output Property** from the right-click menu.

    c. Provide a name for the property in the New Exposed Property dialog box.

    d. Click the REST method wrapper in the canvas, and view the newly exposed property in the Properties pane's **Custom Input/Checkpoints** tab .

    For details, see "Exposing REST Service Properties" on page 1742.

16. ## Edit the prototype service

    To edit a REST service method's properties, select it in the Toolbox pane and choose **Edit Service** from the right-click menu. For details, see the "Add/Edit REST Service Dialog Box" on page 1772.

17. **Resolve conflicts between the prototype and the test step - optional**

    If you created a test step using a template REST method and then changed the template method, the canvas adds an alert icon adjacent to the affected steps. Click the alert icon to open the Resolve REST Method wizard to resolve any conflicts. For details, see the "Resolve REST Method Steps Wizard" on page 1871.

18. **Share the service - optional**

    To add a REST service to the test repository, to make it available for other tests, share it. Select the parent service node in the Toolbox and choose Move to from the right-click menu. For details, see "How to Perform Activity Sharing" on page 1761.

    For an example of creating a REST Service with data driving and data tables, see the *HP Tutorial for UFT* - API Testing.

How to Send a JSON Request to a REST Service

Relevant for: API testing only

This task describes how to send a JSON request for a REST service.

This task includes the following steps:

- "Set the HTTP properties" below
- "Prepare to load the request body" below
- "Load the request body" on the next page
- "Modify the JSON body - optional" on the next page
- "Data drive or link to fields - optional" on the next page

> **Note:** The following tasks show how to send a single JSON request. To create a reusable model, create a prototype. For details, see "How to a Create a REST Service" on page 1746.

1. **Set the HTTP properties**

    > **Note:** If you are working with an API test created in UFT 11.51 or earlier or Service Test 11.51 or earlier, you must expand the REST activity's wrapper and enter these properties in the HTTP Request step found inside the REST activity wrapper.

    In the Properties pane's Input/Checkpoints tab , set the destination **URL** and the **HTTP method**, usually POST or PUT.

2. **Prepare to load the request body**

Open the Properties pane's **HTTP** view . Select a **Body** type **JSON**.

3. ### Load the request body

   Click the **Load JSON** button and locate the .json file. Click **OK**.

4. ### Modify the JSON body - optional

   If you intend to dynamically assign values to the JSON body from a data source, you need to add escape characters. Open the **Text** view of the JSON body and add the escape character, \, for each occurrence of a square or curly bracket ({, }, [, and]). Do not use an escape character for link expressions enclosed by curly brackets. For example:

   ```
   \{"results":
   \[
   \{"name": "John", "id": 873829904, location: "NY"\},
   \{"name": "Linda", "id": 726371109, location: "LA"\},
   \{"name": "Mike", "id": 711029345, location: "NY"\},
   \]
   \}
   ```

   When no links are used, this is not required.

5. ### Data drive or link to fields - optional

   To data drive or link to specific JSON fields, highlight the value in the body text, click the **Link Body** button , and select a data source. For details, see "Select Link Source Dialog Box (API Testing)" on page 1840.

How to Receive a JSON Response from a REST Service

Relevant for: API testing only

This task describes how to receive a JSON response from a REST service. All HTTP Request activities are capable of receiving JSON responses.

This task includes the following steps:

- "Add a request header - optional" on the next page
- "Add checkpoints - optional" on the next page

Note: The following tasks show how to receive a single JSON request. To create a reusable model, create a prototype. For details, see "How to a Create a REST Service" on page 1746.

1. **Add a request header - optional**

   > **Note:** If you are working with an API test created in UFT 11.51 or earlier or Service Test 11.51 or earlier, you must expand the REST activity's wrapper and enter these properties in the HTTP Request step found inside the REST activity wrapper.

   If your server requires you to specify JSON during content negotiation, you need to set the request header. In the Properties pane's Input/Checkpoints tab, click the plus sign to add a **RequestHeader** array element. Add a custom request header named Accept with the value application/json.

2. **Add checkpoints - optional**

   If you require checkpoints to validate the response:

   a. Open the Properties pane's **HTTP** tab. Click in the lower section of the tab, in the checkpoint area.

   b. To validate against regular expressions, set the **Body** type to **Text**. Add regular expressions for each value that you want to validate.

   c. To validate against a JSON response, set the **Body** type set to **JSON**. Click **Load JSON** and locate the .json file with the expected values.

How to Import a Web Application Service

Relevant for: API testing only

This task describes how to import a Web Application service (WADL) into your test.

This task includes the following steps

- "Prerequisite " on the next page
- "Import the WADL document" on the next page
- "Edit the service's properties - optional" on page 1757
- "Use the WADL service's methods" on page 1757
- "Share the service - optional" on page 1757

1. **Prerequisite**

   Before importing, study the structure of your WADL document, as specific elements from the document are imported into the WADL hierarchy inside UFT.

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://wadl.dev.java.net/2009/02">
 <resources base="http://example.com/api">
  <resource path="books">
   <method name="GET"/>
   <resource path="{bookId}">
    <param required="true" style="template" name="bookId"/>
    <method name="GET"/>
    <method name="DELETE"/>
    <resource path="reviews">
     <method name="GET">
      <request>
       <param name="page" required="false" default="1" style="query"/>
       <param name="size" required="false" default="20" style="query"/>
      </request>
     </method>
         <resource path="{index}">
            <method name="GET" id="get index"/>
            <param name="index" style="template"/>
         </resource>
    </resource>
   </resource>
  </resource>
  <resource path="readers">
   <method name="GET"/>
  </resource>
 </resources>
</application>
```

   For details on WADL elements, see http://www.w3.org/Submission/wadl/.

2. **Import the WADL document**

   In the toolbar, click on the **Add REST Service** button and select **Import WADL** or select **Tools > Add REST Service > Import WADL**. In the Choose WADL file dialog box, navigate to the directory in which the WADL is saved, and select the WADL file.

   Note: You can import only WADLs saved on the file system.

The WADL service is imported to the **Local Activities** node of your test with its resources and methods. The WADL service, resource, and method hierarchy is created based on the XML description provided in the WADL file, described below:

XML Element	UFT WADL Activity Hierarchy Element
doc xml:lang="en" title="RestService"	**WADL service name**
resource path="<resource name>"	**WADL resource** **Note:** If multiple resources have the same name, UFT numbers the resources sequentially to differentiate between resources.
method name="<method name>"	**WADL method** ▪ UFT assigns the WADL hierarchy name using the following criteria: ▪ If a method name has an "id" attribute, the name is taken from the value of the "id" attribute. ▪ If a method name does not a have an "id" attribute, then the name is defined as the value of the "method name". For example, if the "method name" is defined as <"method name="GET"/>, UFT defines the method name in the WADL hierarchy as GET Method. The method name always contains the HTTP method as part of its value. This method (GET, POST, PUT, DELETE, TRACE, OPTIONS, or HEAD is also used as the HTTP method in the HTTP tab of the WADL service. For details on the HTTP tab, see "HTTP Tab (Properties Pane - API Testing)" on page 414 **Note:** If there are multiple methods of the same name using the default values, then the methods are defined with increasing sequential numbers.

param name="<parameter name>"	**WADL resource or method parameters.** These parameters are displayed until the Custom Input/Checkpoints tab in the Edit REST Service dialog and in Input/Checkpoints tab for methods on the canvas. If a "param name = <name>" string also contains a "default=<value>" string, the value defined in the XML is displayed with the parameter.
resources base="http://example.com/api"	**WADL Service URL.** This is displayed on the HTTP tab for the service. The URL is also passed to all resources and methods included in the service. For details, see "Passing REST Service Properties" on page 1738. **Note:** You cannot change the URL property for an individual resource or method.

3. **Edit the service's properties - optional**

   To edit the WADL service properties, right-click the service name in the Toolbox and select **Edit Service**.

   In the Edit REST Service dialog box, you can define general properties, parameters, HTTP properties, and HTTP request and response information for the service, resources, and individual methods. For details, see "How to a Create a REST Service" on page 1746.

   For user interface details, see "Add/Edit REST Service Dialog Box" on page 1772.

4. **Use the WADL service's methods**

   After importing a WADL, you have created a prototype for the service, with methods, HTTP properties, and parameters. Drag a method to the canvas to use it in your test.

   Note: You can edit all method properties after dragging the activity to the canvas.

   Caution: If you modify the service's properties after dragging a method to the canvas, the properties of your service will differ from the service's prototype (as defined in the Toolbox). The Resolve REST Steps Wizard is not available to resolve differences for WADL service methods.

5. **Share the service - optional**

   To add a WADL service to the test repository, to make it available for other tests, share it. Select the parent service node in the Toolbox and choose **Move to** from the right-click menu.

For details, see "How to Perform Activity Sharing" on page 1761.

How to Import a Network Capture File

Relevant for: API testing

This task describes how to import a network capture file to your test to create test steps.

This task includes the following steps:

- "Create a capture file" below
- "Prerequisite - study the structure of your network capture file" below
- "Import the network capture file" on the next page
- "Edit the step properties - optional " on the next page
- "Run the step - optional" on the next page

1. ### Create a capture file

   Use a network capture program (also known as a sniffer) to create a capture file containing a log of network activity for your application or Web service.

   > **Note:** It is strongly recommended that you capture network traffic only on your application or Web service during the network capture session to prevent the creation of invalid or unneeded activities in your test. Many network capture tools capture all network traffic on the computer where they are installed, including network traffic unrelated to your application or Web service.

2. ### Prerequisite - study the structure of your network capture file

   The structure of your file depends on the type of file you import. UFT supports .libpcap/pcap and .har network capture files.

   - **For .pcap files:** UFT creates test steps based on the TCP network traffic. You can see the input and checkpoint values by viewing the TCP request and response information for a TCP stream.

     > **Note:** You must use a network capture program to view the network capture traffic for your .pcap file.

   - **For .har files:** Using a text editor, you can view the Request and Response information in the JSON hierarchy of the file.

Note: UFT creates a test activity based on its recognition of the TCP stream in the following ways:

- If UFT recognizes the TCP stream as being compatible with or matching an already existing Web Service method, UFT creates a **Web Service** step.
- If UFT recognizes the TCP stream response as a SOAP network transaction, it creates a **SOAP Request** step.
- If UFT does not recognize the TCP stream as being similar to an existing Web Service step or a SOAP request network transaction, it creates an **HTTP Request** step.

3. **Import the network capture file**

   a. In UFT, with an API test open or selected, select **Tools > Import Network Capture File**. The "Import Network Capture Dialog Box" (described on page 1781) opens.

   b. In the Import Network Capture Dialog Box, select the file containing your network capture data.

   Note: UFT supports only .pcap and .har network capture files.

   c. View the file details in the **Info Pane** of the Import Network Capture Dialog Box. If there are errors in your file, return to your network capture tool and fix the errors.

   d. If you want UFT to create checkpoints for your test steps based on the response sections of the network capture file, select the **Create checkpoints from response** option.

   e. Click **Import**. If your network capture file contains many transactions, UFT displays the progress of your import.

   Note: To stop an import, click **Cancel** in the import progress window. All test steps created prior to your cancellation are removed from the test.

   UFT creates Web Service test steps, SOAP Request test steps, or HTTP Request test steps for each of the transactions contained in your network capture file.

4. **Edit the step properties - optional**

   In the Properties pane, select the properties you want to edit and enter the values as needed. For details on populating property values, see "How to Assign Data to API Test/Component Steps" on page 1819.

5. **Run the step - optional**

   To verify that the step runs correctly after the import and step creation, right-click the step in the canvas and select **Run Step**.

You can view the step results and the request and response information for the test step in the Run Step Results pane.

How to Import and Create a .NET Assembly API Test Step

Relevant for: API testing only

This task describes how to create a test step for verifying the functionality of a .NET assembly.

This task includes the following steps:

- "Prerequisite " below
- "Open the Import .NET Assembly dialog box" below
- "Select a GAC assembly - optional" below
- "Import a .NET assembly" below
- "Add a .NET assembly step" on the next page
- "Add properties - optional" on the next page
- "Customize the code - optional" on the next page

1. ### Prerequisite

   Prepare a .dll assembly file and store it in a location accessible from your machine.

2. ### Open the Import .NET Assembly dialog box

   Click the **Import .NET Assembly** button on the toolbar. For user interface details, see "Import .NET Assembly Dialog Box" on page 1774.

3. ### Select a GAC assembly - optional

   Each computer where the common language runtime is installed has a machine-wide code cache called the GAC (Global Assembly Cache). The GAC stores assemblies specifically designated to be shared by several applications on the computer.

   In the **GAC** tab, select one or more GAC assemblies and click **Select** to add them to the **Selected References** list.

4. ### Import a .NET assembly

   Click the **.NET Assembly Browser** tab. Locate the assembly .dll or .exe file. Select the file and click **Select** to add it to the **Selected References** list. Click **OK** to add the .NET assemblies to the Toolbox pane.

> **Tip:** After importing the .NET assembly, you should save the test. Not saving the test leads to unexpected behavior with the autocomplete functionality when adding an event handler to a .NET assembly test step.

5. ### Add a .NET assembly step

   Expand the **.NET Assemblies** category in the Toolbox pane and drag a .NET activity onto the canvas.

6. ### Add properties - optional

   a. In the Properties pane's Input/Output tab, click the **Add Input/Output Property** button ✚ Add... ▾ to define new input and output properties.

   b. In the **Add Input/Output Property** dialog box, specify a property name and data type.

      - To add a simple type property, select a type from the drop down.

      - To add an advanced type, click **Advanced** and select a type. All .NET types and .dll files referenced by the test, and all types from the imported assemblies, are available. To filter the types, enter a keyword into the **Type** field.

   c. Click **OK** to add the property. Repeat this for all of the required input and output properties.

7. ### Customize the code - optional

   In the Properties pane, open the **Events** tab ⚡ . Double-click in the ExecuteEvent's **Handler** column. The TestUserCode.cs tab opens.

   Edit the Todo area. You can access input and output properties that you defined earlier, using autocompletion.

   The comments indicate the syntax to use for accessing the properties. For example:

```
/// Use this.CodeActivity4 to access the CodeActivity4
/// Activity's context, including input and output properties.
public void CodeActivity4_OnExecuteEvent(object sender, STActivityBaseEventArgs args)
{
this.CodeActivity4.Input.Property1...
...
}
```

How to Perform Activity Sharing

Relevant for: API testing only

This task describes how to save activities to a repository, update them, and view their properties.

This task includes the following steps:

- "Connect to ALM" below
- "Set up the repository paths" below
- "Import a WSDL or create a REST service" below
- "Move the activity to a repository" below
- "View the service properties - optional" below
- "Refresh the activity - optional" on the next page
- "Update the activity - optional" on the next page
- "Add the activity to a new test - optional" on the next page

1. Connect to ALM

If you want to work with a repository on ALM, connect to the desired ALM server. For details, see the "ALM Connection Dialog Box" on page 737.

2. Set up the repository paths

a. Select **Tools > Options > API Testing** tab **> General** node.

b. In the Activity Repositories section, click the **Browse** button and navigate to the location in the file system or in ALM.

c. Click **OK**.

3. Import a WSDL or create a REST service

Import a WSDL file or create a REST service method. The Toolbox pane shows the services in the Local Activities node. By default, the test stores these activities in its EmbeddedJavaResources subfolder.

4. Move the activity to a repository

Right-click the service node, and select **Move to > File System Activities** or **ALM Activities**. This moves the service from Local Activities to **File System Activities** or **ALM Activities** in the Toolbox pane. The service is also removed from the test's EmbeddedJavaResources subfolder and placed in the repository folder. The next time you create a test, these activities will be accessible from the **File System Activities** or **ALM Activities** nodes.

5. View the service properties - optional

Right-click the service's parent node, and select **Properties**. The Properties window shows the service's significant properties. For details see "Web Services Properties Dialog Box" on page 1769.

6. **Refresh the activity - optional**

   To reload the WSDL from its stored location, right-click the service node, and select **Refresh** . This is useful in cases when the WSDL is stored in a shared location and may have been updated by another user.

7. **Update the activity - optional**

   To reimport the WSDL from its original location, for example to revert back to the original version, right-click the service node and select **Update WSDL**, or click the **Update WSDL** button in the Toolbox pane .

8. **Add the activity to a new test - optional**

   Open a new test. Expand the **File System Activities** or **ALM Activities** node, select the desired service or method, and drag it onto the canvas.

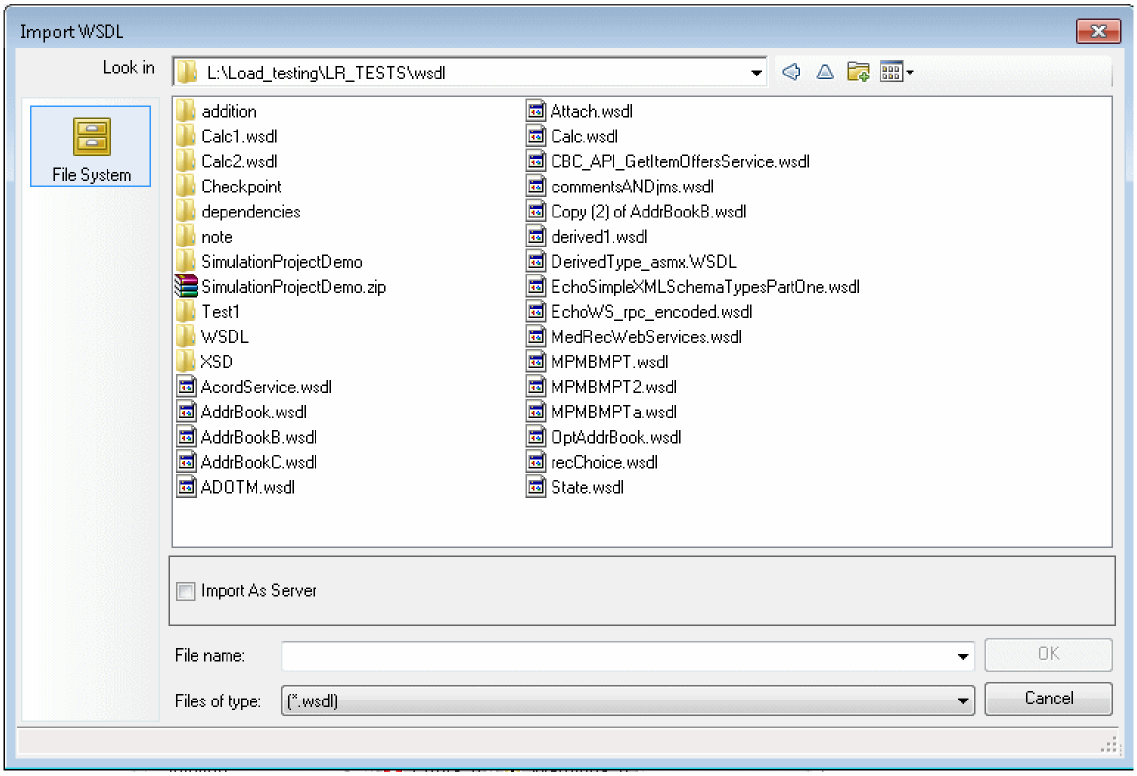
   You can also remove shared activities from the toolbox and re-add them. For details, see "Toolbox Pane User Interface (API Testing) " on page 514.

References

Select WSDL Dialog Box

Relevant for: API testing only

This dialog box enables you to import WSDLs from a file system or Service Test Management, the ALM extension for working with application components.



Import WSDL

Look in: L:\Load_testing\LR_TESTS\wsdl

□ Import As Server

File name:

Files of type: (*.wsdl)

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Select **Import WSDL > Import WSDL from File or ALM Application Component**.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 1744

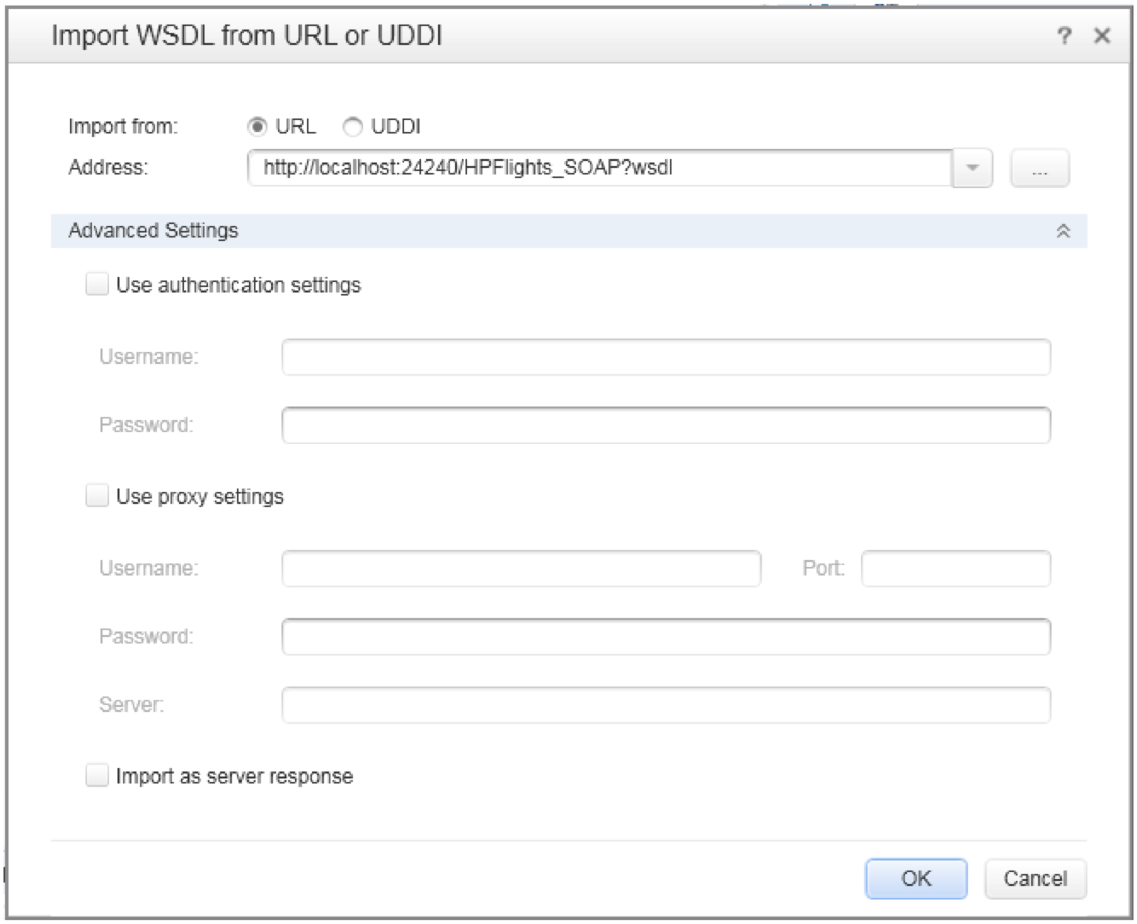
The following elements are included:

UI Elements	Description
File System	Displays WSDLs in the file system.
ALM Application Components	Displays WSDLs in the ALM repository. **Note:** Available only if there is an open connection to ALM with the Service Test Management extension.
Import as Server Response	Imports the WSDL as a server response. Its methods will be server responses—not requests. This is useful for asynchronous type Web services.
File name	The contract name: • **File System.** The file name of the WSDL. • **ALM Application Components.** The name of the application component or service.
Files of type	The document type: **WSDL** extension or **All files**.

Import/Update WSDL from URL or UDDI Dialog Box

Relevant for: API testing only

This dialog box enables you to import WSDLs using a URL or UDDI or update an existing WSDL from a new location.



Import WSDL from URL or UDDI

Import from: ● URL ○ UDDI

Address: http://localhost:24240/HPFlights_SOAP?wsdl

Advanced Settings

□ Use authentication settings

Username:

Password:

□ Use proxy settings

Username: Port:

Password:

Server:

□ Import as server response

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Select **Import WSDL > Import WSDL from URL or UDDI.**
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 1744

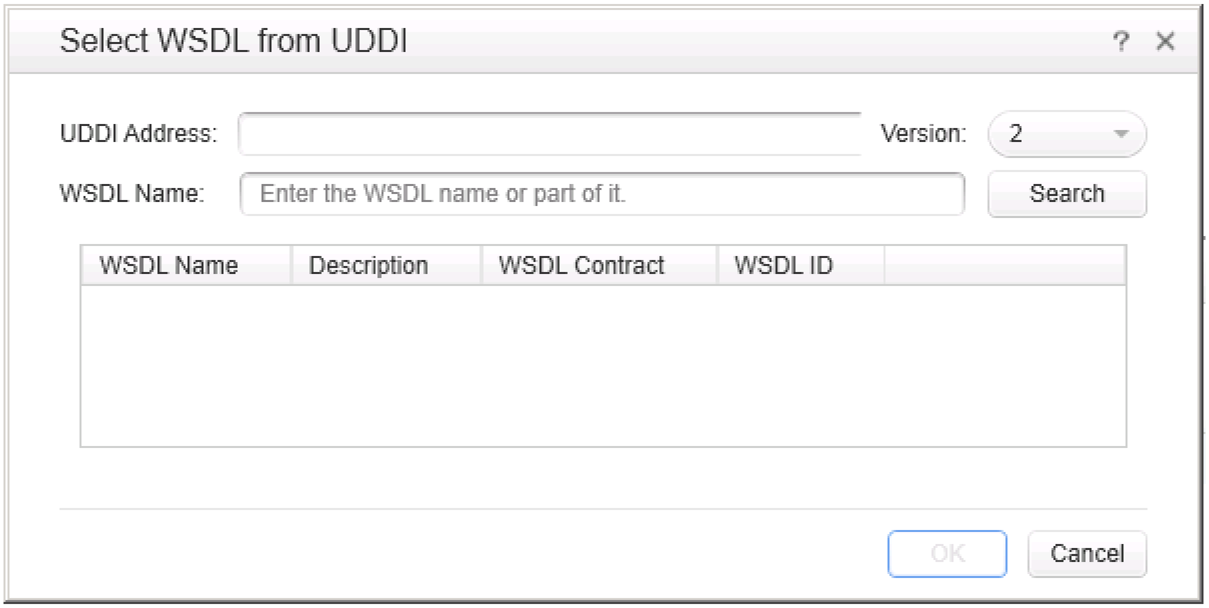
The following elements are included:

UI Elements	Description
...	**Browse**. - For **URL** imports: Opens a browser to allow you to navigate to a URL. - For **UDDI** Imports: Opens the "Select Service from UDDI Dialog Box".
Address	The URL or UDDI path of the WSDL.
Advanced Settings	Expands the dialog box to show the authentication and proxy settings.
Use authentication settings	The authentication settings by which to access the WSDL: **Username** and **Password**.
Use proxy settings	The authentication settings for the proxy server hosting the WSDL: **Server**, **Port**, **Username**, and **Password**.
Import as Server Response	Imports the WSDL as a server response. Its methods will be server responses—not requests. This is useful for asynchronous type Web services.

Select Service from UDDI Dialog Box

Relevant for: API testing only

This dialog box enables you to select and import WSDLs from a UDDI (Universal Description, Discovery, and Integration) server.



Select WSDL from UDDI

UDDI Address:

Version: 2

WSDL Name: Enter the WSDL name or part of it.

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Select **Import WSDL > Import WSDL from URL or UDDI.** 3. Select **Import from: UDDI**. 4. Click the … button.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 1744
See also	"Import/Update WSDL from URL or UDDI Dialog Box" on page 1766

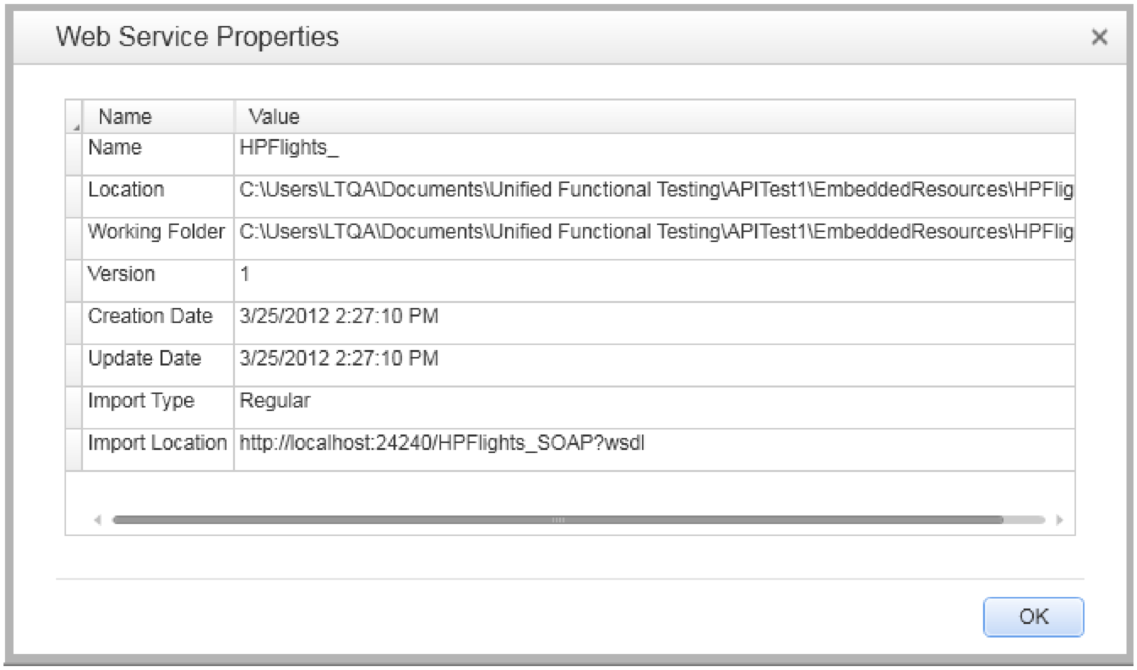
The following elements are included (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
UDDI Address	The name or IP address of the UDDI server inquiry API. **Example:** http://mysite.com:8080/uddi/inquiry
Version	The UDDI version.
<list of services>	A list of the services that match the filter criteria. The grid shows the following information: • **Service Name** • **Description** • **Service Contract** • **Service ID**
Service Name	A string in the WSDL name by which to filter the list.
Search	Retrieves the list of services that match the string in the **Service name** field. If no string is specified, it retrieves all available services.

Web Services Properties Dialog Box

Relevant for: API testing only

This dialog box displays the properties of imported Web services.



Name	Value
Name	HPFlights_
Location	C:\Users\LTQA\Documents\Unified Functional Testing\APITest1\EmbeddedResources\HPFlig
Working Folder	C:\Users\LTQA\Documents\Unified Functional Testing\APITest1\EmbeddedResources\HPFlig
Version	1
Creation Date	3/25/2012 2:27:10 PM
Update Date	3/25/2012 2:27:10 PM
Import Type	Regular
Import Location	http://localhost:24240/HPFlights_SOAP?wsdl

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Import a WSDL. Select **Import WSDL > Import WSDL from URL or UDDI**. 3. Select the parent node of the WSDL in the Toolbox pane. 4. Select **Properties** from the context menu.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 1744

The following elements are included:

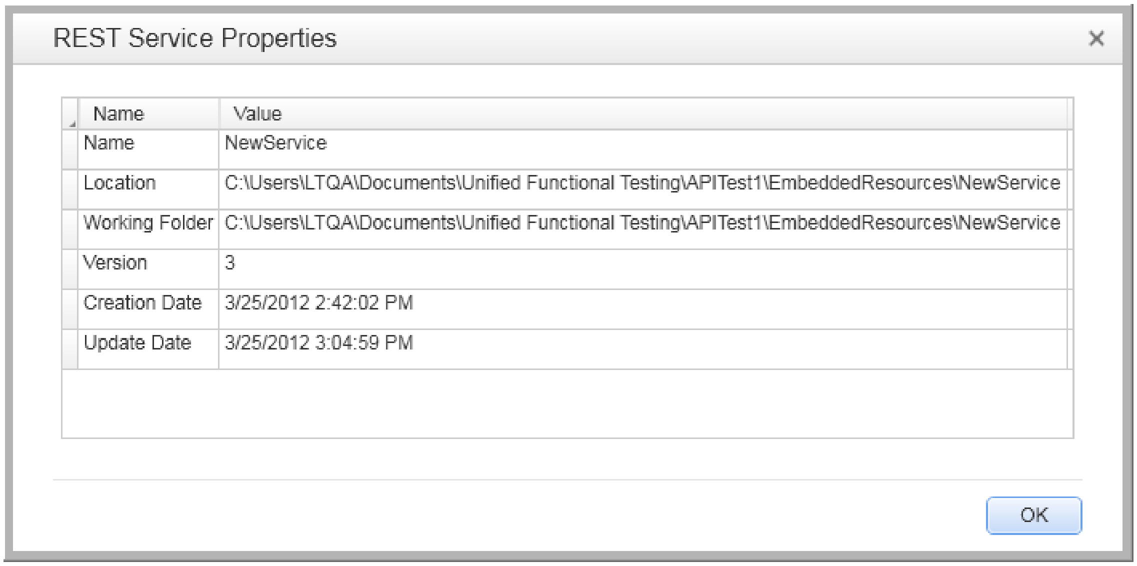
UI Elements	Description
Name	The name of the Web service as defined in the WSDL file.

UI Elements	Description
Location	The current location of the WSDL. If it was updated from a location other than the original, it will indicate the new location.
Working Folder	A folder containing a local, working copy of the WSDL. For Web services moved to the shared repository, this is a temporary folder.
Version	The WSDL version.
Creation Date	The date and time in which the WSDL was first imported.
Update Date	The date and time of the latest update. If no updates were done, this value is the same as the **Creation Date**.
Import Type	The way in which the WSDL was imported: **Regular** or **As Server**. For details about importing the WSDL as a server response, see the "Select WSDL Dialog Box" on page 1764.

REST Services Properties Dialog Box

Relevant for: API testing only

This dialog box displays the properties of REST services.



Name	Value
Name	NewService
Location	C:\Users\LTQA\Documents\Unified Functional Testing\APITest1\EmbeddedResources\NewService
Working Folder	C:\Users\LTQA\Documents\Unified Functional Testing\APITest1\EmbeddedResources\NewService
Version	3
Creation Date	3/25/2012 2:42:02 PM
Update Date	3/25/2012 3:04:59 PM

To access	1. Do one of the following: - Ensure that an API test or component is in focus in the document pane. - In the Solution Explorer, select an API test or component. 2. Create a REST Service or import a Web Application service. For details, see "Add/Edit REST Service Dialog Box" on the next page. 3. Select the parent node of the service in the Toolbox pane. 4. Select **Properties** from the context menu.
Relevant tasks	"How to a Create a REST Service" on page 1746

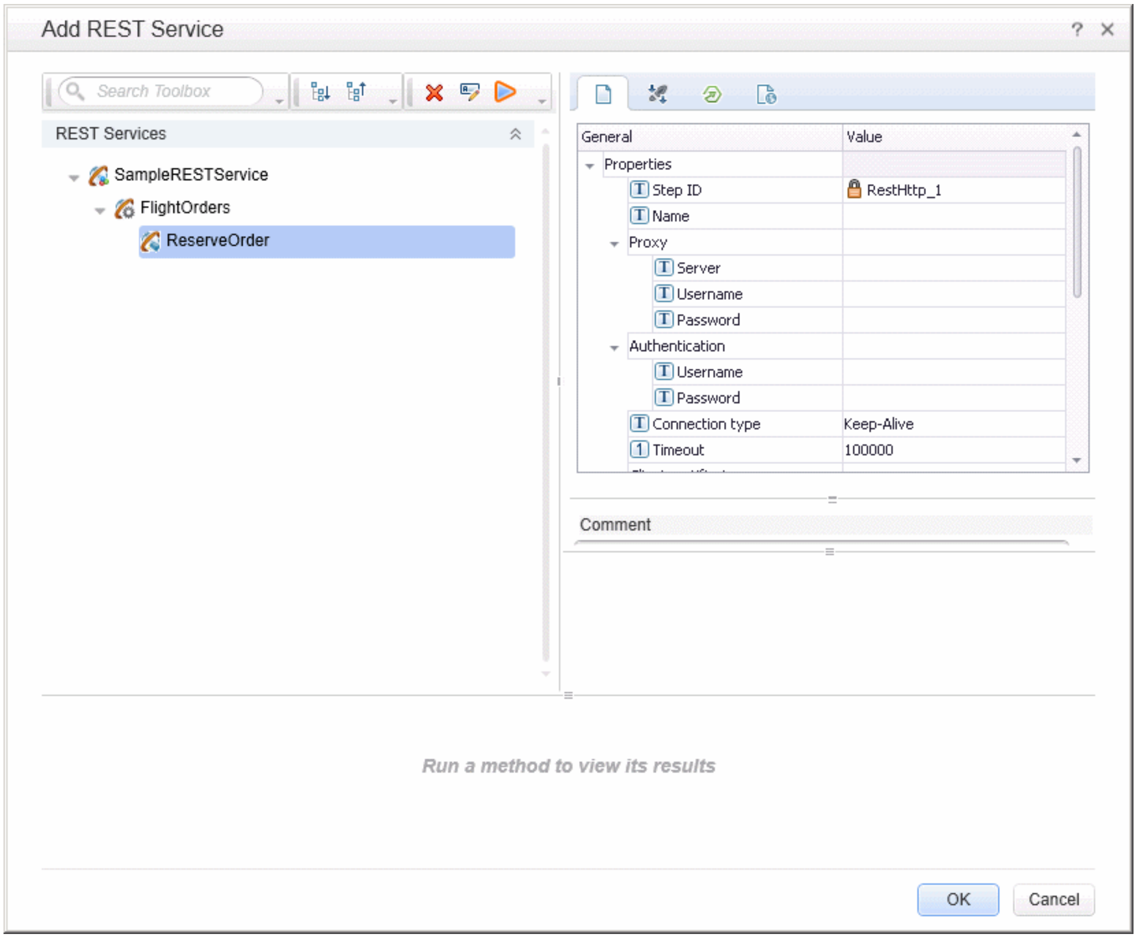
The following elements are included:

UI Elements	Description
Name	The name of the REST service as defined in the REST Service Designer.
Location	The current location of the REST service files.
Working Folder	The folder containing a local, working copy of the test. By default, this is <Test_ Folder>\EmbeddedResources\ NewService.
Version	The current version of the REST service.
Creation Date	The date and time in which the service was created.
Update Date	The date and time of the latest change and refresh. If no updates were done, this value is the same as the **Creation Date**.

Add/Edit REST Service Dialog Box

Relevant for: API testing only

This dialog box enables you to define a new REST service.



General	Value
Properties	
Step ID	RestHttp_1
Name	
Proxy	
Server	
Username	
Password	
Authentication	
Username	
Password	
Connection type	Keep-Alive
Timeout	100000

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Do one of the following: ▪ Click the **Add REST Service** button on the toolbar and select **REST Service Editor** or **Import WADL**. ▪ Select a REST Service or Web Application service node in the Toolbox pane, and choose **Edit Service** from the right-click menu.
Relevant tasks	"How to a Create a REST Service" on page 1746

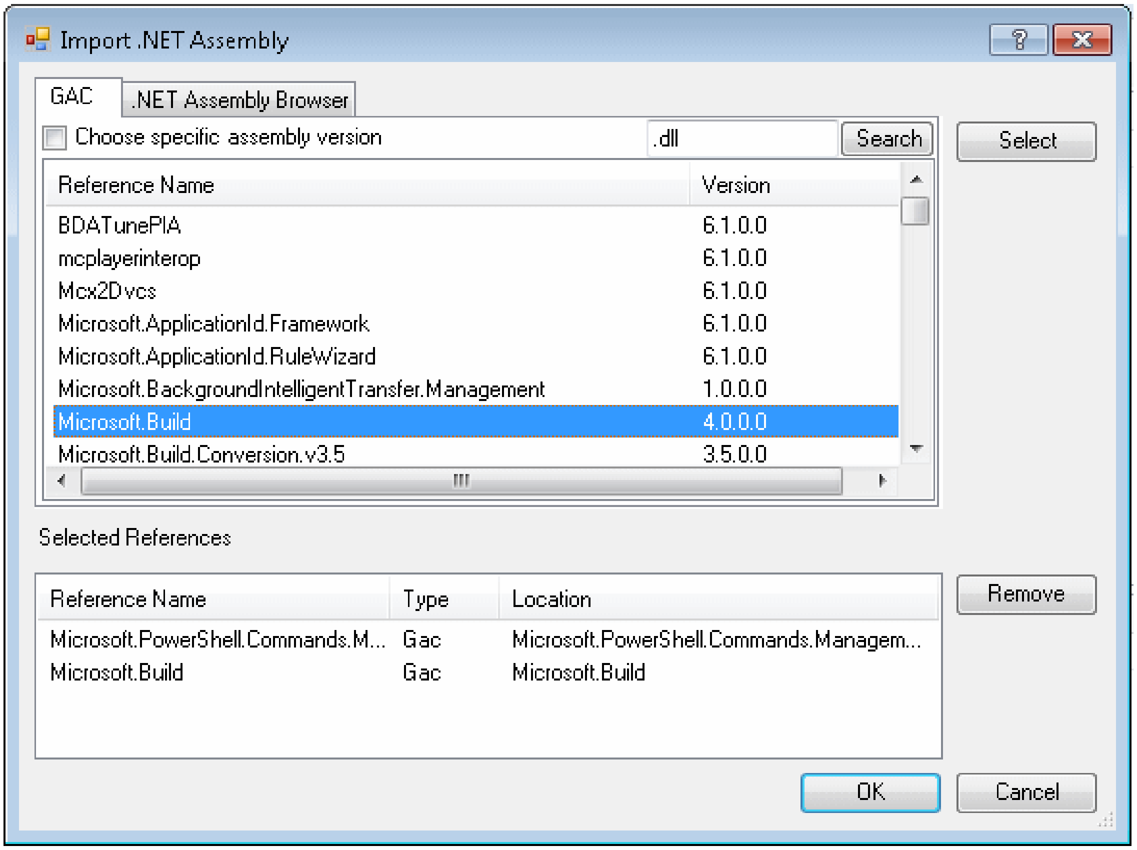
Some user interface elements are available only when you select a specific node. The user elements are (unlabeled elements are shown in angle brackets):

UI Elements	Description
	Expand All. Expands all of the nodes of the parent REST service.
	Collapse All. Collapses all of the children nodes of the parent REST service.
	Add Method. Adds a method to the selected REST service node.
	Add Resource. Adds a resource to the selected REST service or resource.
✖	**Delete.** Deletes the selected REST service, resource, or method.
	Rename. Renames the selected REST service, resource, or method.
▶	**Run Method.** Runs the selected method and shows the results in the lower part of the pane. **Note:** Available only when you select a method.
<properties pane>	The properties of the REST service methods. For details, see the "Properties Pane" on page 385.
<REST Services tree>	Displays a list of the REST services that match the filter condition.
<results pane>	The results of the Run Method command. This section also shows the property values and the response.
Filter box	Filters the display by the entered text.

Import .NET Assembly Dialog Box

Relevant for: API testing only

This dialog box enables you to import .NET assemblies in order to use them as activities from the Toolbox pane.



□ Choose specific assembly version

.dll

Reference Name	Version
BDATunePIA	6.1.0.0
mcplayerinterop	6.1.0.0
Mcx2Dvcs	6.1.0.0
Microsoft.ApplicationId.Framework	6.1.0.0
Microsoft.ApplicationId.RuleWizard	6.1.0.0
Microsoft.BackgroundIntelligentTransfer.Management	1.0.0.0
Microsoft.Build	4.0.0.0
Microsoft.Build.Conversion.v3.5	3.5.0.0

Reference Name	Type	Location
Microsoft.PowerShell.Commands.M...	Gac	Microsoft.PowerShell.Commands.Managem...
Microsoft.Build	Gac	Microsoft.Build

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Click the **Import .NET Assembly** button on the toolbar.
Important Information	After importing the .NET assembly, you should save the test. Not saving the test leads to unexpected behavior with the autocomplete functionality when adding an event handler to a .NET assembly test step.
Relevant tasks	"How to Import and Create a .NET Assembly API Test Step" on page 1760

The following elements are included:

UI Elements	Description
GAC tab	Displays a list of the GAC (Global Assembly Cache) assemblies. **Choose specific assembly version.** Shows all versions of the GAC assemblies as separate entries. This enables you to add a specific version of a GAC assembly.
.NET Assembly Browser tab	Enables you to browse for an assembly to import.
Select	Adds the selected assemblies to the **Selected References** list.
Selected References	A list of the GAC assemblies that you selected. The grid shows the following information: • **Reference Name.** The name of the reference as it will appear in the Toolbox pane. • **Type.** GAC or Assembly • **Location.** For GAC types, the class; For Assembly types, the full path of the DLL.
Remove	Removes the selected assemblies from the **Selected References** list.

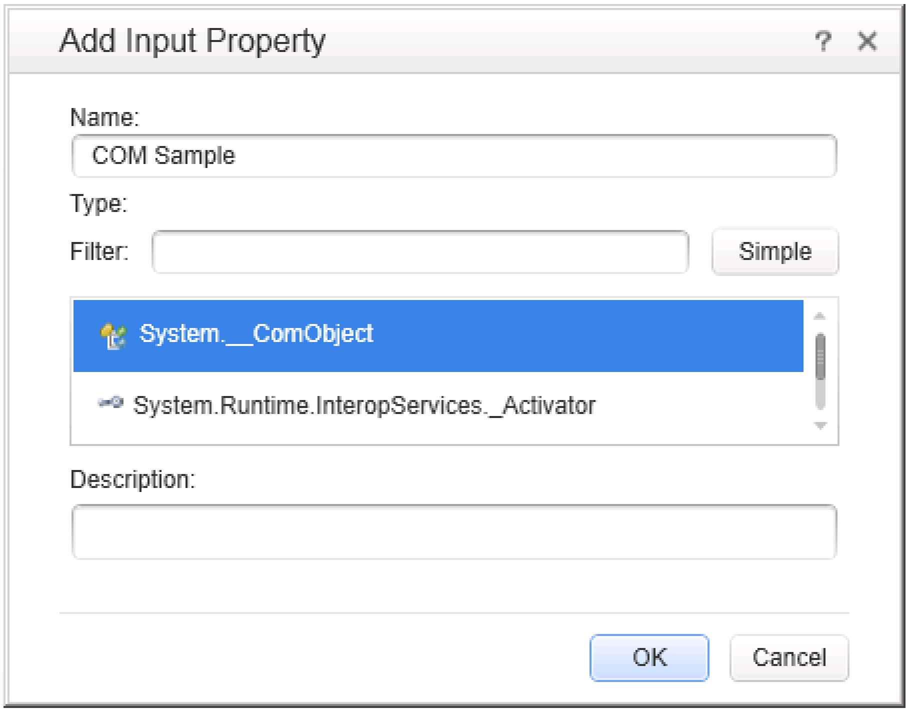
Add Input/Output Property/Parameter Dialog Box (API Testing)

Relevant for: API testing only

This dialog box enables you to define custom input or output properties for the current step or input or output parameters for a test.

- For adding a property for a Custom Code or .NET Assembly step, see "Add Property for Custom Code or .NET Assembly Activities" on the next page.
- For adding a test property, see "Add Input/Output Parameter for Test Settings" on page 1777.
- For adding a property for a REST activity step, see "Add Input/Output Property for REST Service Steps" on page 1779

Add Property for Custom Code or .NET Assembly Activities



Add Input Property

Name: COM Sample

Type:

Filter:

System.__ComObject

System.Runtime.InteropServices._Activator

Description:

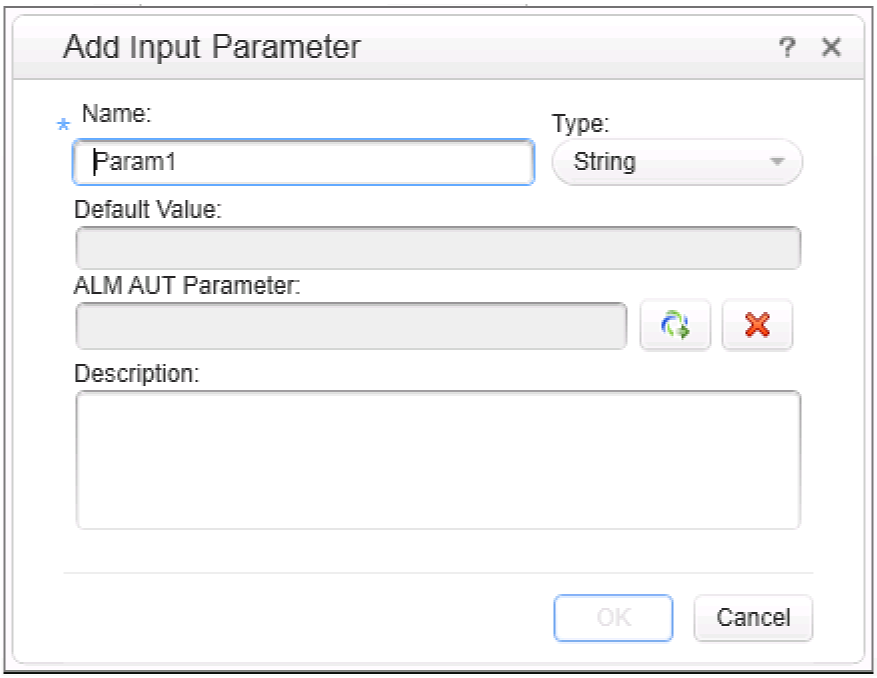
To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Add a Custom Code or .NET Assembly activity to the canvas. 3. Select the Properties pane's **Input/Checkpoints** tab. 4. Click the **Add** button and select one of the following: ▪ **Add Input Property** ▪ **Add Output Property**
Relevant tasks	• "How to Import and Create a .NET Assembly API Test Step" on page 1760 • "Create a Custom Code activity" on page 1600 • "How to Define API Test Properties or User/System Variables" on page 141

The following elements are included (unlabeled elements are shown in angle brackets):

UI Elements	Description
Name	A name for the property.
Type	For **Simple** types, a drop down list of simple data types, such as String, Integer, and so forth.
Filter	For **Advanced** types, a textual expression by which to filter the displayed assemblies. For example, to show only those assemblies that begin with the **system** prefix, enter system.
Simple	Shows the simple data types, such as String, Integer, and so forth.
Advanced	Shows an extensive list of types, including all .NET types and DLLs referenced by the test, and all types from the imported assemblies.
<list of properties>	A list of the types, including .NET types, types from DLLs referenced by the test, and all types of the imported assemblies.
Description	A description of the property.

Add Input/Output Parameter for Test Settings

The image below shows the dialog used to add an input parameter. This example also shows the options available when adding a test parameter to a test stored on an ALM server with Lab Management.



Add Input Parameter

Name: Param1

Type: String

Default Value:

ALM AUT Parameter:

Description:

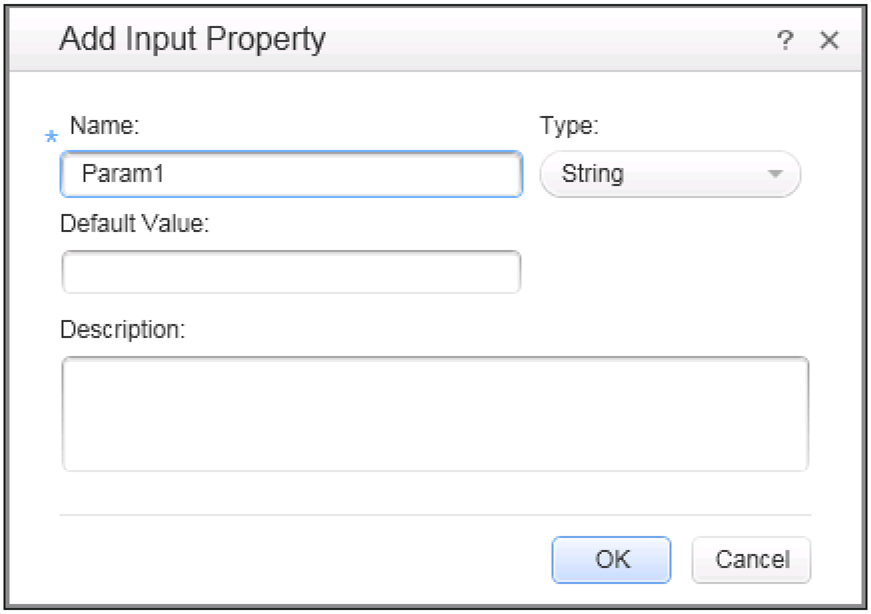
To access	1. Create or open an API test or action. 2. Select the **Start** or **End** steps in the canvas. 3. Click the Properties pane's **Test Input/Output Parameters** tab. 4. Click the **Add** button and select one of the following: ▪ **Add Input Parameter** ▪ **Add Output Parameter** (not relevant for ALM AUT Parameters)
Relevant tasks	"How to Run a Test Using Server-Side Execution" on page 727.
See also	• "Running Tests in Server-Side Execution" on page 713 • "Select AUT Parameter Dialog Box" on page 742

The following elements are included:

UI Elements	Description
Name	A name for the parameter. A default name for the parameter is provided.
Type	A drop down list of simple data types, such as String, Integer, and so forth. **Note:** For parameters linked to ALM AUT parameters, the type will always be String.
Default Value	A default value for the parameter, used during the run session if no other value is provided for the parameter. **Note:** For ALM AUT parameters, you can set this value by selecting **Copy default value (override)** in the "Select AUT Parameter Dialog Box" (described on page 742).
ALM AUT Parameter	The path to an ALM AUT Parameter from the **Lab Resources > AUT Environment** module in ALM Lab Management. You cannot edit the text in this box. Use the **Select ALM application parameters** to enter an AUT parameter. **Note:** Available only if UFT is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.

UI Elements	Description
⟳	**Select ALM application parameters.** Opens the Select AUT Parameter dialog box for selecting a parameter from an AUT environment defined in the **Lab Resources > AUT Environment** module. For details, see "Select AUT Parameter Dialog Box" on page 742. **Note:** Available only if UFT is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
✖	**Remove link to ALM application parameters.** Removes the linking of the selected parameter from ALM. This parameter will now be an ordinary test parameter. **Note:** Available only if UFT is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
Description	A description of the parameter.

Add Input/Output Property for REST Service Steps


Add Input Property

* Name: Param1

Type: String

Default Value:

Description:

To access	1. Do one of the following: ■ Ensure that an API test or component is in focus in the document pane. ■ In the Solution Explorer, select an API test or component. 2. Open the Add REST Service Dialog box . 3. Create a REST method, service, or resource. 4. Click the **Custom Input/Checkpoints** tab in the right pane . 5. Click the **Add** button and select one of the following: ■ **Add Input Property** ■ **Add Output Property** (for REST methods only)
Relevant tasks	"How to a Create a REST Service" on page 1746.
See also	"Add/Edit REST Service Dialog Box" on page 1772

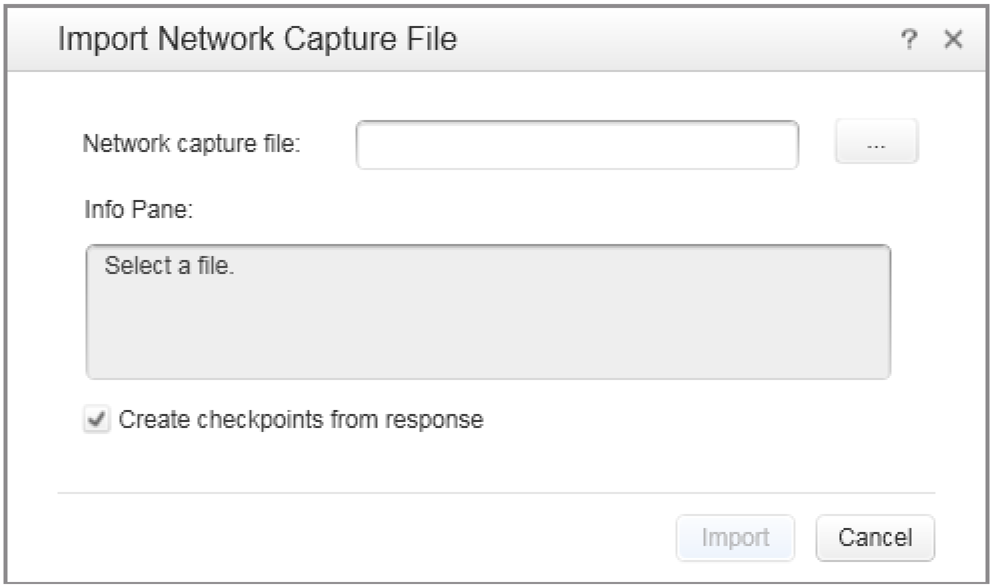
The following elements are included:

UI Elements	Description
Name	A name for the property.
Type	A drop down list of simple data types, such as String, Integer, and so forth.
Description	A description of the property.

Import Network Capture Dialog Box

Relevant for: API testing only

This dialog box enables you to import the contents of a network capture file. After you import the file to your test, UFT creates test steps for the transactions included in your file.



Import Network Capture File

Network capture file:

Info Pane:

Select a file.

☑ Create checkpoints from response

To access	1. Create, open, or select an API test. 2. Select **Tools > Import Network Capture File**.
Important Information	• Only .pcap and .har network capture files are supported. • If your file contains errors, you cannot import the file and the **Import** button is disabled. • Steps created by importing a network capture file are not stored in the Toolbox pane. If you need to add the steps to your test again, you need to import the network capture file again.
Relevant tasks	"How to Import a Network Capture File" on page 1758
See also	"Network Capture Activities" on page 1736

The following elements are included (unlabeled elements are shown in angle brackets):

UI Elements	Description
Network Capture file	The path to the file to import. **Note:** You can only select network capture files stored on the file system.
...	**Browse.** Enables you to navigate to the file to import.
Info Pane	Displays details about the selected file, including number of network transactions contained in the file or errors in the file. **Note:** The maximum number of transactions allowed is 100.
Create checkpoints from response	Creates checkpoint values for your test steps based on the information detailed in the response of the .pcap or .har file. You view the response of your file differently based on whether your network capture file is a .pcap or a .har file.: • **For .pcap files:** Open the file in a network capture program and view the response information contained in a TCP stream. • **For .har files:** Open the file in a text editor and navigate to the response section of the desired HTTP transaction.

Troubleshooting and Limitations - Local Activities

Relevant for: API testing only

This section describes troubleshooting and limitations for working with Web and REST services.

This section includes the following:

Web Services

- You cannot import WSDL files with names that are restricted by the Windows operating system. This list includes: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9.

  Workaround: Rename the WSDL file before the import.

- The following WSDLs are not supported:
  - WSDL version 2.0
  - WSDLs containing a <appInfo> element.
  - RPC Encoded WSDLs configured as Asynchronous Web services.
  - WSDLs authenticated by HTTP digest on Apache servers.

- When opening many tests in the same solution using one or more of the same Web services, UFT may develop a memory leak.

  Workaround: Move the Web service to the file system or ALM repository. To do this, import a Web service and then, in the **Toolbox** pane, right-click the Web Service activity and select **Move to > File System Activities** or **ALM Activities**.

- For a Web service imported as a server response:
  - RPC type WSDLs cannot be imported as a server response. If you attempt to update a service from an RPC encoded WSDL, it will create a duplicate entry.

- If you end the UFT.exe process during the listening stage, after the binding was added, the binding will not be removed from the system.

  Workaround: Remove the binding manually using a utility such as httpcfg.exe or netsh.exe.

- When working with SSL, certificates from a file are not supported. If you move the test to another machine, the certificate will not be saved with the test.

  Workaround: Add the certificate to the local machine store before the listener starts, and remove it at the end of the listening process.

REST Services

- Importing a schema to a REST, HTTP, or SOAP checkpoint may remove the links of the input properties.

- XPath checkpoints are not supported for HTTP and REST activities.

- If you link between input and output properties in the same REST method and then delete the input property without removing the link, the link expression still remains in the output property. If you save the test in this state, you may be unable to reopen it.

  Workaround: Delete the link explicitly either before or immediately after deleting its source property.

- When defining a REST method prototype in the Add/Edit REST Service dialog box - in order to use the Trim, Ignore, or Stop test options, select the check box in the **Validate** column, in the Checkpoints pane.

- When running a REST method using the **Run Step** command, checkpoints of dynamic property values in the method linked to other property values (input or output) are ignored.

Web Applications (WADL)

The following elements are not supported when importing your WADL. UFT does not import these elements into the WADL hierarchy inside your test:

- Grammars element

- Resource_type element

- Link element

> **Note:** Any child elements of these elements are also ignored by UFT and not added to your WADL inside the test.

In addition, if you use the href attribute to link to other elements, you must refer to an element in the same WADL file. Linking between WADL files is not supported.

For more details on WADL elements, see http://www.w3.org/Submission/wadl/

Network Capture Activities

Steps added to your test from a network capture file do not include:

- Security settings for Web service calls and SOAP requests
- Web authentication information for any type of step
- Cookies data for HTTP request
- Attachments sent as part of the Web Service call

.NET Assemblies

Importing or adding references to 64-bit .NET assemblies is not supported.

Chapter 57: Reusable Actions in Service Test

Relevant for: API testing only

This chapter includes:

Concepts

Actions in API Testing - Overview

Relevant for: API testing only

Actions are sequences of operations that you perform within the context of a test, consisting of either one or multiple steps. Calls to actions placed in your test enable you to test a specific area or function of your application. In this way, your test can check the user interface and the application functionality, such as opening a flight reservation system using based on a Web Service.

Actions also allow you call a repeated activity, such as logging in to a Web site multiple times. Each time the action is called, the test calls the action steps, its properties, and its data. Instead of recreating the process and entering its data each time you call the action, the call to an action refers to the original action instead of having to define a new action.

An action contains its own test coding, including all of the steps in that action and all necessary objects for its execution.

When you insert a call to an action into your test, the test adds the action to the Solution Explorer. As a result, you can reuse the action multiple times within a test or call it from another test.

Actions can also be saved as business components to be used for BPT.

When to Use Actions

Actions are useful when you need to repeat a specific activity multiple times. For example, if you are testing your Web service which requires a login, you can insert a call to a log-in action multiple times instead of inserting a new log-in step each time it is needed.

Actions also enable you to parameterize repeated steps within your test. For example, if you want to test your Web service for multiple users, you can call an action connected to a data source with user information for multiple users. The call to an action gives you the ability to provide data for a specific step or feature to ensure it works for multiple scenarios or users. Thus, you can not only test the functionality of the processes of your application, but you can also test that the individual elements within that process work correctly.

Actions also enable you to modify a specific element within an application without affecting the test. When the call to action is inserted in the test of a larger process, if the application element changes, the entire test does not. The action properties are updated to reflect the change in the application element. However, the application process test steps do not need to be modified.

Calling Actions from Other Tests

You can call both actions defined within your current test or defined in other tests. When you call an action from another test, by default its data is read-only. Using the Property View, you can enable the action's data for editing. For details, see "How to Use Actions in an API Test" on page 1793.

Nesting and Repeating Actions

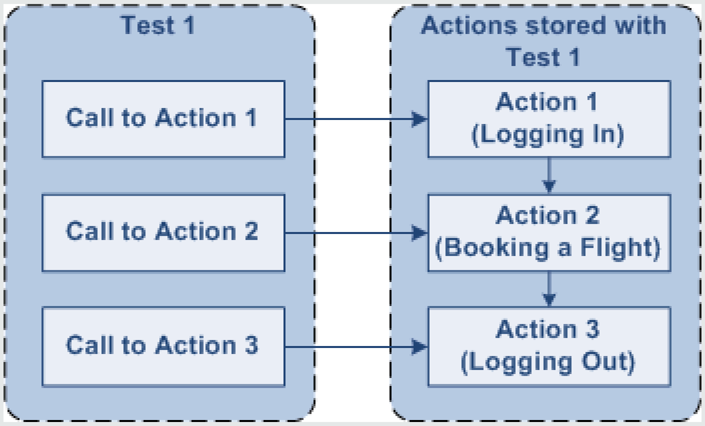
You can nest additional actions within existing ones. This allows you to create specialized actions and call them at the desired point within your test.

You can also set the loop properties for individual actions, independent of the loop properties in the main Test Flow. You click in the action's flow, and specify the loop properties in the Property View.

Example

Suppose you want to test a Web service which enables customers to book a flight and purchase tickets. This requires you to test several processes and features: request a Web service, log in, choose flights, process customer information, output a complete reservation, and log out. Testing this service also requires you to test both application processes and user interface elements. Furthermore, you might want to test the capability of a user, such as a travel agent to use the service to make multiple reservations.

Your test might look similar to this model:



Action Placement

Relevant for: API testing only

UFT provides reusable actions that can be called multiple times within a test or by other tests. These actions make it easier to maintain your tests, because when an object or procedure in your application changes, the modified element is updated in all calls to the action.

Note: This is different from GUI Test (formerly QuickTest Professional) actions, which allow you to indicate whether the action should be reusable or not.

Calls to Existing Actions

Relevant for: API testing only

When you plan a test or multiple tests, you may realize that each test requires identical, repeated activities, such as logging in. Instead of inserting these steps three times in separate places or tests (and adding checkpoints, parameters, and data) separately, you can create an action one time and store it with the test. You then populate the action with activities from the Toolbox. After you are satisfied with the action you have created, you can also call it from other tests.

If you call an action from an external test that has data assigned to its properties, you can indicate whether you want the data from action's data sheet to be imported as a local, editable copy, or whether you want to use the (read-only) data from the original action.

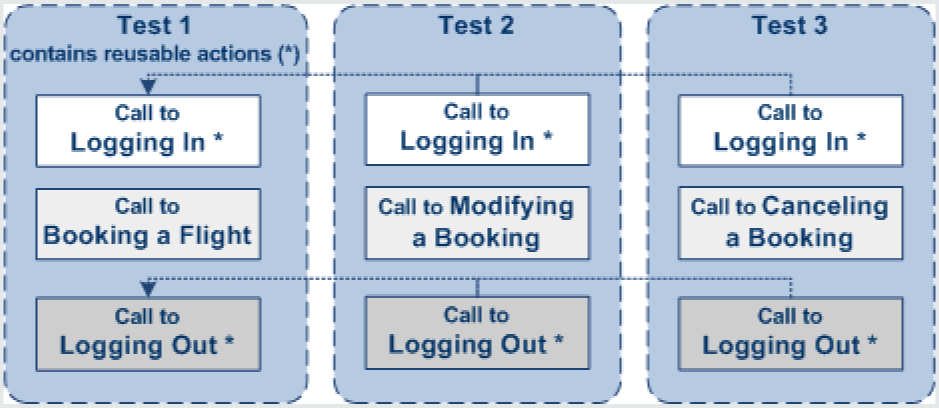
To modify the steps of an action from an external test, you must open the test in which the action is stored and make your modifications there. The modifications apply to all tests that call that action.

For more details, see "How to Use Actions in an API Test" on page 1793.

Example

Suppose you want to create the following three tests for flight reservation--booking a flight, modifying a reservation, and deleting a reservation. While planning your tests, you realize that for each test, you need to log in and log out of the site.

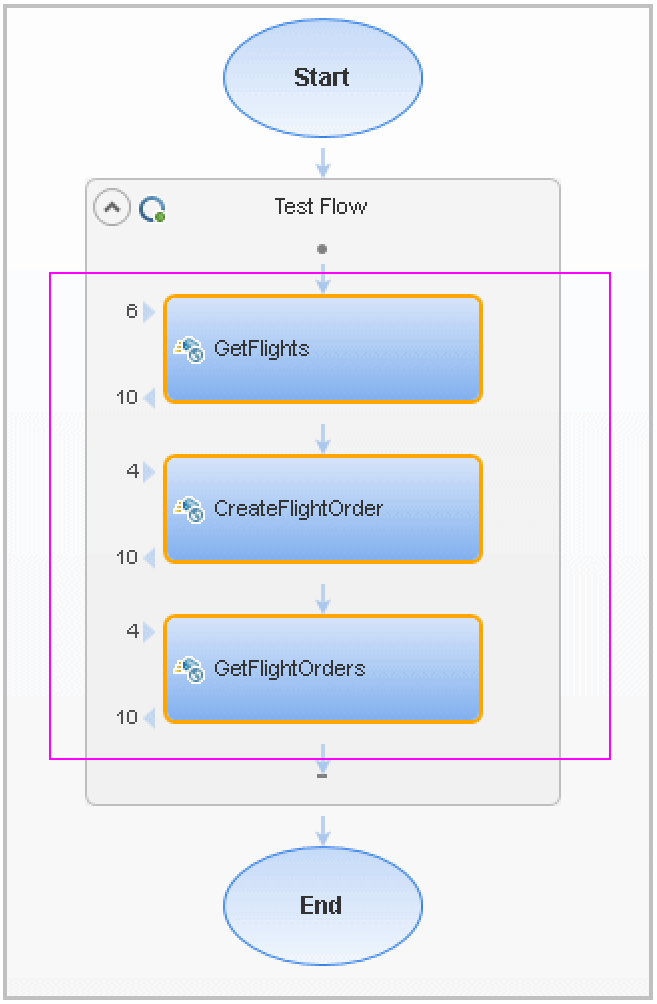
Test 1 would contain three actions (Logging In, Booking a Flight, and Logging Out). Test 2 would contain three actions (Logging In, Modifying a Booking, and Logging Out). Test 1 would contain three actions (Logging In, Cancelling a Booking, and Logging Out). The Logging In and Logging Out actions are defined in the same test, and called as external actions by all three tests.



Combining Steps into Actions

Relevant for: API testing only

Multiple steps can be combined together to constitute an action. For example, when designing a test to check your Web service for booking flights, you may find that certain actions and processes are repeated multiple times. You can select these steps on the canvas and group them in an action. You can also select multiple steps using the standard Windows multiple selection methods, with the **SHIFT** and **CTRL** keys.



After selecting the steps you copy/cut and paste them into the action's tab. Your test can then call the action, and apply loop behavior to the action as a single unit.

For more details, see "How to Use Actions in an API Test" on page 1793.

Actions Using the Data Table

Relevant for: API testing only

You can use the Data pane tables to provide property values for your actions. Each action uses its own data sources—not the test's data source.

> **Tip:** If you want to use the same data for the test and in a specific action, you must define the data source path twice, once for the test and once for the action. See "Using Data in Your API Test or Component Steps" on page 1801.

By default, data from actions called from other tests, are not visible in your current test. You can expose the data by enabling the **Allow other tools to override the data** option in the action’s test. As a result, you can view the data in the Data Pane. It is marked with an icon indicating that this is data from an action.

An **Update** option allows you to reload the action's data in your test when it changes in its original location. This only applies to data that you did not make editable. For details, see "How to Use Actions in an API Test" on page 1793.

> **Note:** You can switch data from read-only to editable or vice versa. However, when changing from editable to read-only, any changes made to the data are lost.

If an action is called repeatedly in the test, only one set of the action's data is displayed in the Data Pane. If you make changes to the data at the action's original location and it is not marked as editable, then the changes are applied to all calls to this action.

For details on how UFT handles data in a test, see "Using Data in Your API Test or Component Steps" on page 1801.

Considerations for Working with Actions

Relevant for: API testing only

Inserting Actions

You should consider using actions if:

- You intend to use an identical or virtually identical procedure or action in more than one test.
- You want to edit the data in a specific call to an action that you call multiple times within your test.

Naming Actions

You may want to rename the actions in your test with descriptive names to help you identify them. It is also a good idea to add detailed action descriptions. This facilitates inserting actions from one test to another. You can rename an action by right-clicking on the action in the Solution Explorer and selecting **Rename**.

When renaming an action, make sure to use the following naming conventions:

- Cannot begin or end with a space
- Cannot exceed 1,023 characters
- Cannot contain the following characters: **\ / : * ? " < > | % ' ! { }**

Tasks

How to Use Actions in an API Test

Relevant for: API testing only

This task describes the different operations you can perform to use actions in your test.

This task includes the following steps:

- "Insert a call to a new action" below
- "Call an existing action or a test" below
- "Set action properties" on the next page
- "Copy or move steps" on the next page
- "Remove a call to an action" on the next page
- "Delete an action" on the next page
- "Enable an action's data for editing when called by another test" on page 1795
- "Override data from an action called by another test" on page 1795

Insert a call to a new action

1. Select **Design > Call to New Action**.
2. Specify the action name and description. Click **OK** to add the call to the action to the canvas.
3. In the Solution Explorer, double-click the action to open its canvas.
4. Add steps to the action. Drag activities from the Toolbox onto the canvas.

> **Note:** You can insert a call to another action, from an existing action call.

Call an existing action or a test

1. Drag an **HP Automated Testing Tools > Call API Action** or Test or **Call GUI Action or Test** activity onto the canvas.
2. Open the Properties pane (**CTRL+ALT+P**).
3. In the Input/Checkpoints tab, click the **Select Action or Test** button.
4. Select the desired test and action. Click **OK**.

Set action properties

To set an action call's properties, you must set the values for each of the steps separately.

1. Select an activity within the action whose properties you want to set.

2. In the Properties tab, provide values or data drive the properties. For details, see the "Properties Pane Tabs" on page 403.

3. The property values that you set will be the default values used when you add a call.

   > **Note:** If you reload the service using the **Refresh** button, the new version may add or remove properties. Properties that remain unchanged, will retain the default values.

Copy or move steps

Copying or moving steps is ideal if you want to copy or move existing steps into another location in the Test Flow or to an action.

1. Make sure the actions into which you want to copy or move the steps, are open. To open an action, double-click on it in the Solution Explorer or select **Open** from the action's context menu in the test.

2. Select one or more steps that you want to copy or move. To select multiple steps, do one of the following:

   - Drag the mouse pointer around the steps. A transparent rectangle is be drawn over the selected area.

   - Press **SHIFT** while selecting the beginning and ending steps to be grouped. All steps between the endpoints are grouped.

3. Select **Copy** or **Cut** from the context menu.

4. Place the cursor in the desired location. Select **Paste** from the right-click menu.

Remove a call to an action

In the test canvas, select the call to the action and press **DELETE** or right-click and select **Delete**

Delete an action

In the Solution Explorer, right-click the action and select **Delete**. The action is removed from the Solution Explorer and all calls to the action are removed. Edited data from the removed step is moved to the main node of the Data Pane as described in "Actions Using the Data Table" on page 1791.

Enable an action's data for editing when called by another test

To view or override an external action's data, you must expose it in its original location. This only applies to Excel type data sources, and only available for tests—not business components.

1. Define data for the action that you will be calling. Assign the data manually or use data driving. For details, see "How to Assign Data to API Test/Component Steps" on page 1819.

2. In the Data Pane, select the data source's node.

3. Open the Properties pane (**CTRL+ALT + P**) and select the **Allow other tools to override the data** option. The Data pane creates a local copy of the data which can be edited.

Override data from an action called by another test

1. Make sure the action that you want to call has its data exposed. For details, see the above step.

2. Open the test in which you want to add the call to the action.

3. In the **HP Automated Testing Tools** node, drag a **Call API Action or Test** or a **Call GUI Action or Test** activity onto the canvas.

4. In the Properties pane Input/Checkpoints tab, click the **Select Action or Test** button.

5. In the Select Action or Test dialog box, browse for the test and select the desired action. Click **OK**.

6. Open the Properties pane and select the **Use a local, editable copy** option.

7. Edit the data tables in the Data pane.

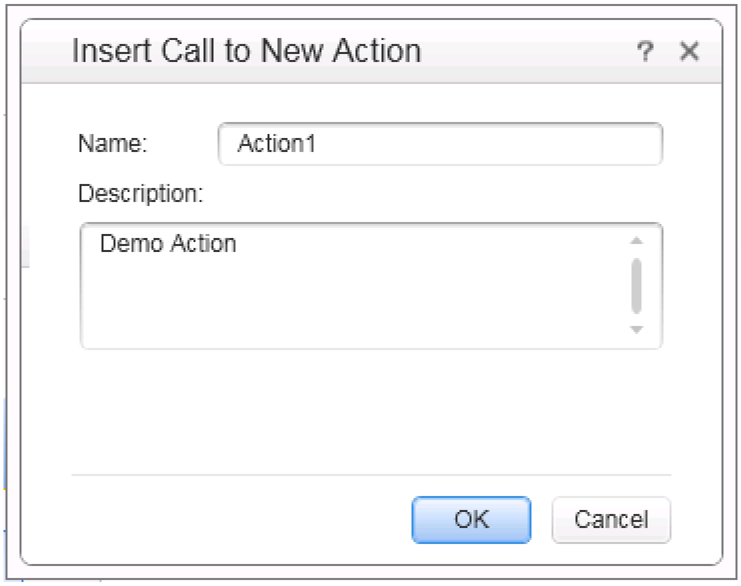
Reference

Insert Call to New Action Dialog Box (API Tests)

Relevant for: API testing only

For GUI tests, see "Insert Call to New Action Dialog Box (GUI Testing)" on page 913.

This dialog box enables you to create a new action for your test.



Name: Action1

Description: Demo Action

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Do one of the following: ▪ **Design > Call to New Action** ▪ Click the **Insert Call to New Action** button
Relevant tasks	"Insert a call to a new action" on page 1793

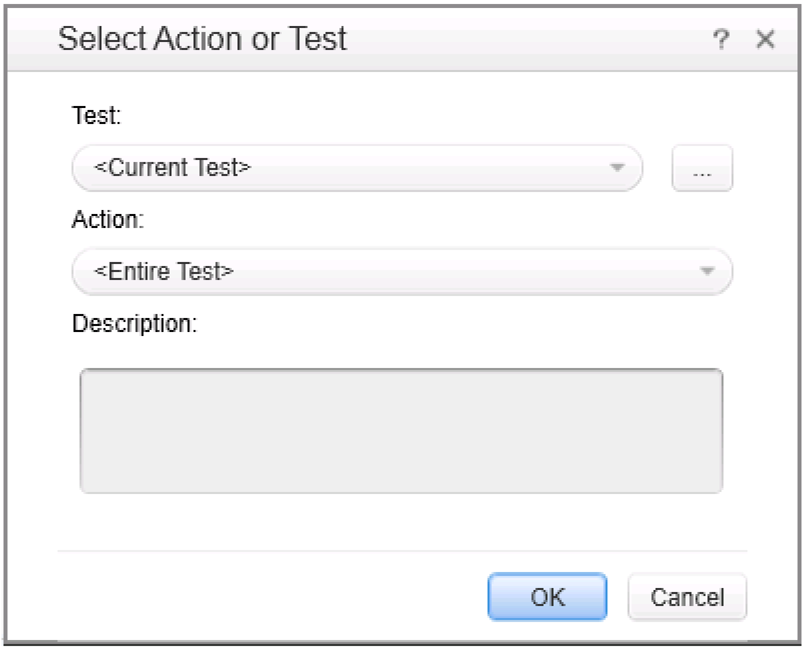
User interface elements are described below:

UI Element	Description
Name	A name for the action as it will appear in the Solution Explorer and canvas.
Description	A detailed description of the action.

Select Action or Test Dialog Box

Relevant for: API testing only

This dialog box enables you to add a call to a copy of an action, or a call to an existing action within your test.



Select Action or Test

Test: <Current Test>

Action: <Entire Test>

Description:

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Drag an **HP Automated Testing Tools > Call API Action or Test**or **Call GUI Action or Test** activity onto the canvas. 3. Open the Properties pane. 4. Click the **Select Action or Test** button.
Relevant tasks	"Call an existing action or a test" on page 1793

User interface elements are described below:

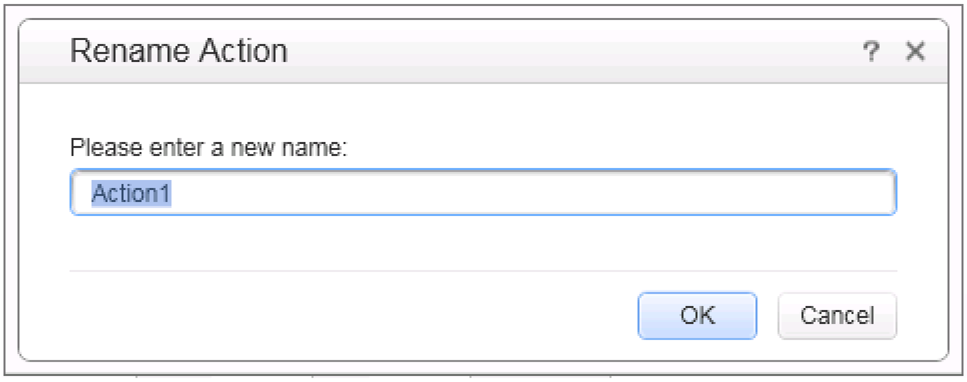
UI Element	Description
Test	The name of the test from which you are calling the action.

UI Element	Description
...	**Browse.** This allows you to browse your test library to find a test.
Action	A drop down list of the actions available for the selected test. You can view a list of the available actions in the Solution Explorer.
Description	The original description given to the action (read-only).

Rename Action Dialog Box

Relevant for: API testing only

This dialog box enables you to rename an action.



Please enter a new name: Action1

To access	Right-click on an API action name in the Solution Explorer.
Relevant tasks	"How to Use Actions in an API Test" on page 1793
See also	"Naming Actions" on page 1791

User interface elements are described below:

UI Element	Description
New Name	A new name for the action.

Actions Context Menu

Relevant for: API testing only

The context menu for action steps is described below:

UI Element	Description
Cut	Cuts the action from the current flow, and places it on the clipboard.

UI Element	Description
Copy	Copies the selected action to the clipboard.
Delete	Removes the selected action from the current flow.
Open	Opens a tab for the selected action. This allows you to add steps and set properties for the action. **Note:** For external actions, stored in other tests, a message window asks if you want to add the external test to your solution. This creates a local copy of the action allowing you to modify its data.

Chapter 58: Data Usage in API Tests

Relevant for: API testing only

For details about the GUI testing Data Table Explorer pane, see "Data Pane" on page 221.

This chapter includes:

Concepts

Using Data in Your API Test or Component Steps

Relevant for: API testing only

When you create test steps, you assign input and checkpoint property values for the test steps. Sometimes you enter the property values manually. Other times you link these property values to data.

In UFT, you can link property values to data in a number of ways:

- **Link to a data source.** You can connect your test steps to an Excel file, an XML data source, a database, or a locally-created data table. These data sources are associated with your test in the Data pane. UFT then pulls the property values from the values specified in the Data pane. This enables you to run the test step using different sets of property values, thereby testing how your test steps (and by extension, your application) performs with varying data values.

  For more details on linking your test steps to a data source, see "Linking Property Values to a Data Source" on page 1803. For details on associating a data source with your test using the Data pane, see "How to Add Data Sources to an API Test" on page 233.

- **Link to another test step:** In some cases, your application passes a value between multiple processes in the course of its normal operation. Therefore, in your test, you can link the output of one step to the input of another step. This mimics the way that your application passes values between application processes, and allows your test to more accurately assess the application's performance.

  For details, see "Linking Property Values to a Data Source" on page 1803.

- **Link to multiple sources.** You can also link your property values to multiple sources by creating a custom expression that combines manual property value input, values from a data source, and input from multiple test steps. As a result, if an application process receives its input from both a data source and the previous step, your test can reflect the same property/parameter input.

  For more details, see "Linking Property Values to Multiple Sources of Data" on page 1809.

- **Link to a test variable value.** In addition to linking to a data source value or another test step, you can provide the property values by linking to a test variable value or a user-defined variable. This enables you to use other custom values for properties in your test.

For task details on using data with test steps, see "How to Assign Data to API Test/Component Steps" on page 1819.

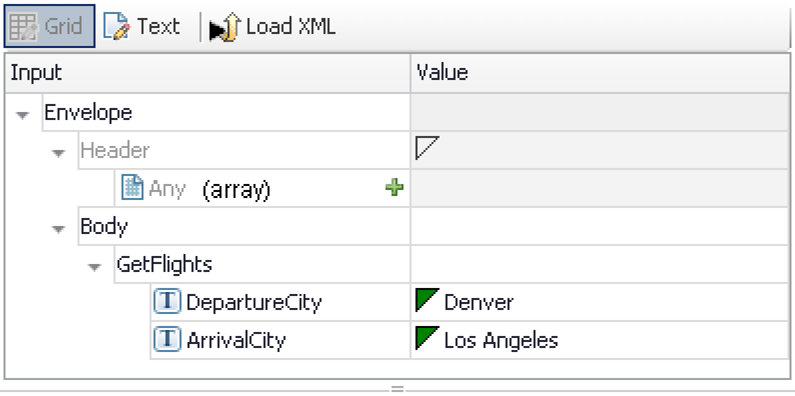
In addition, if you would like to use data for all the properties in your test, you can data-drive all the properties of a selected step. Data-driving is the mechanism by which UFT automatically creates data expressions that refer to data in a new, editable table. For details, see the "Data Driving Dialog Box" on page 1834.

Populating Property Values Manually

Relevant for: API testing only

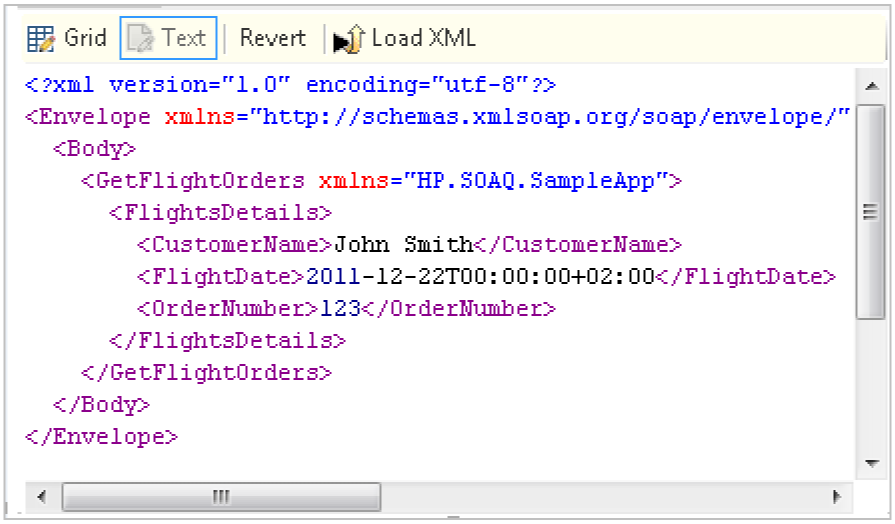
When you enter property values for your test step, the basic way to do this is to provide manual values for each property.

In the "Parameters/Checkpoints Tab (Properties Pane - API Testing)", you can view and edit the values of each input and checkpoint property, both for simple properties and arrays.



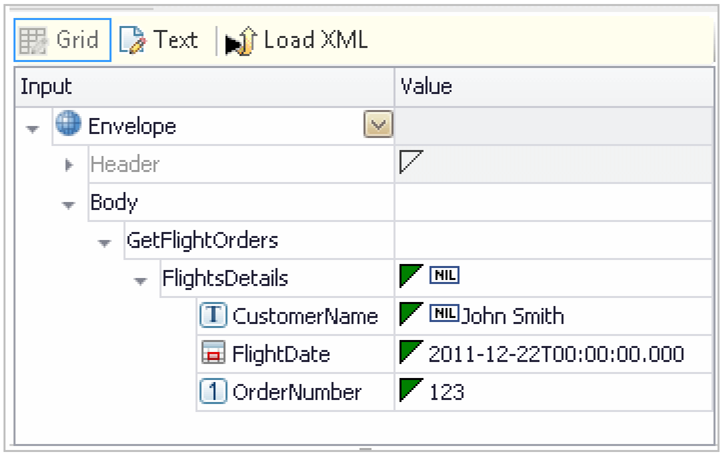
Input	Value
Envelope	
Header	
Any (array)	
Body	
GetFlights	
DepartureCity	Denver
ArrivalCity	Los Angeles

For test steps that use XML and JSON (such as Web Service calls and XML/JSON activities), you can also manually edit the values in the XML/JSON text mode.



```
<?xml version="1.0" encoding="utf-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
  <Body>
    <GetFlightOrders xmlns="HP.SOAQ.SampleApp">
      <FlightsDetails>
        <CustomerName>John Smith</CustomerName>
        <FlightDate>2011-12-22T00:00:00+02:00</FlightDate>
        <OrderNumber>123</OrderNumber>
      </FlightsDetails>
    </GetFlightOrders>
  </Body>
</Envelope>
```

Any changes you make are then reflected in the Input/Checkpoints property grid:



Input	Value
Envelope	
Header	
Body	
GetFlightOrders	
FlightsDetails	NIL
CustomerName	NIL John Smith
FlightDate	2011-12-22T00:00:00.000
OrderNumber	123

In addition, you can manually populate the checkpoint values by loading values that were obtained from an earlier run using the **Load from Replay** mechanism.

Linking Property Values to a Data Source

Relevant for: API testing only

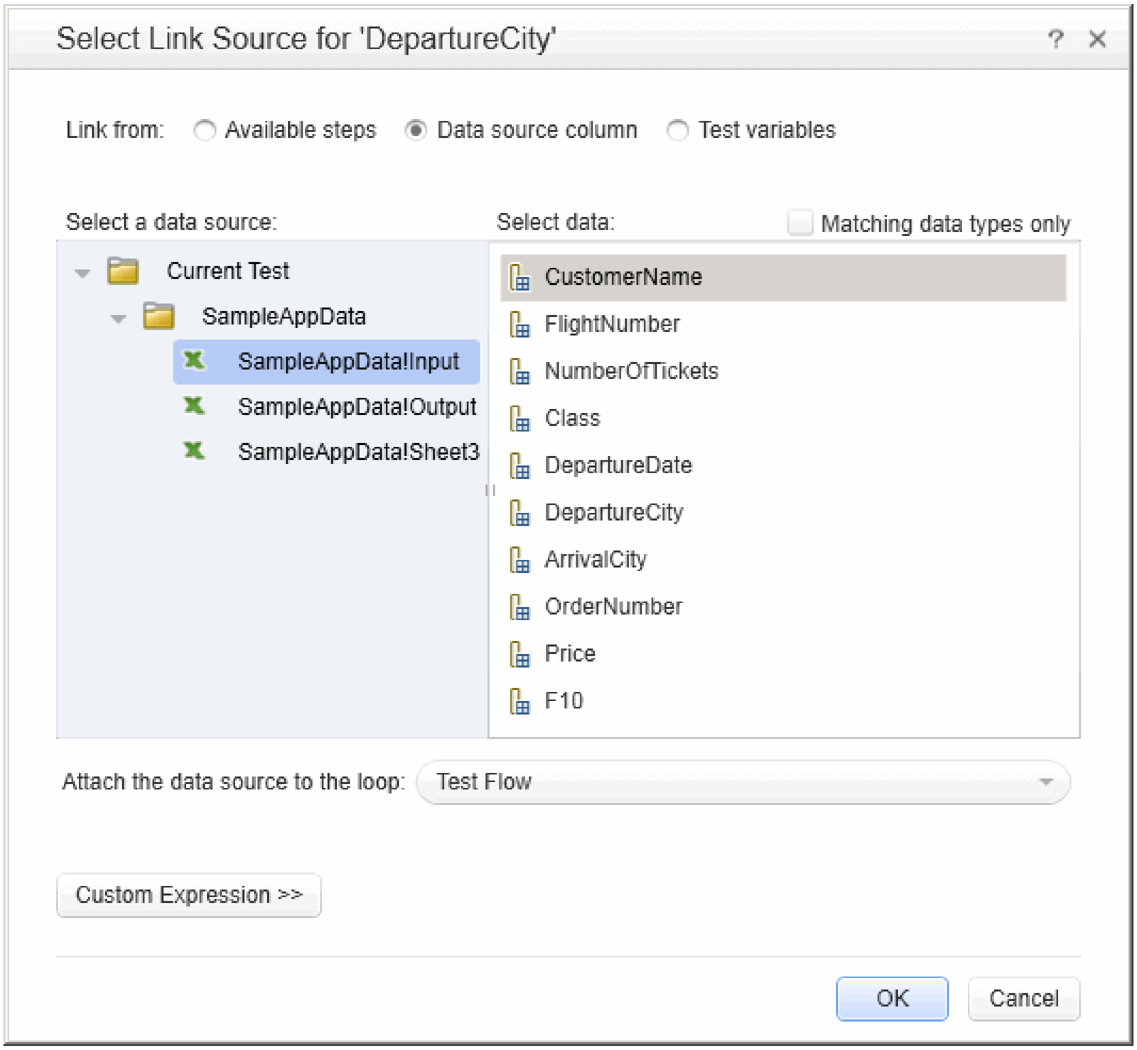
You can link the test input and checkpoint property values to a data source. In this way, you can test how a particular test runs with varying input and checkpoint properties.

> **Note:** Before you can link your property values to a data source, you must import the data source into your test in the Data pane. For details, see "Data Pane Overview" on page 222.

You can link your test steps to multiple types of data:

- Excel files
- XML files
- Database values
- Local table values

The image below shows an example of connecting a test step to an Excel data source using the Select Link Dialog Box:



Select Link Source for 'DepartureCity'

Link from: ○ Available steps ● Data source column ○ Test variables

Select a data source:

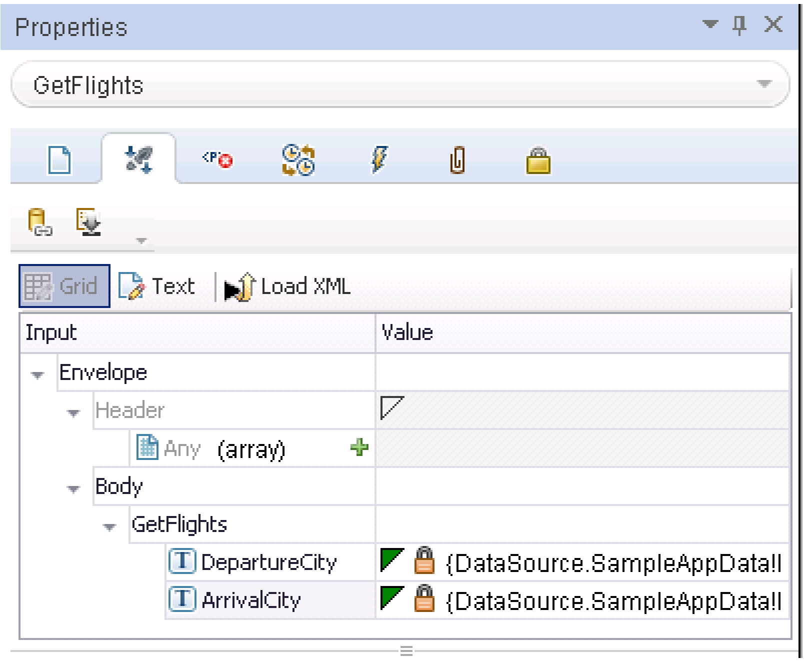
- Current Test
  - SampleAppData
    - SampleAppData!Input
    - SampleAppData!Output
    - SampleAppData!Sheet3

Select data: □ Matching data types only

- CustomerName
- FlightNumber
- NumberOfTickets
- Class
- DepartureDate
- DepartureCity
- ArrivalCity
- OrderNumber
- Price
- F10

Attach the data source to the loop: Test Flow

Once you have linked the step to a specific value in the data source, the Value column for the property name in the Properties pane reflects this link:



Input	Value
Envelope	
Header	
Any (array)	
Body	
GetFlights	
DepartureCity	{DataSource.SampleAppData!I
ArrivalCity	{DataSource.SampleAppData!I

When the test runs, the property values are provided from the associated source.

For task details on how to link property values to a data source, see "Link your test step to a data source" on page 1819. For details on the Select Link Source dialog box, see "Select Link Source Dialog Box (API Testing)" on page 1840.

Creating a Query to Retrieve Database Tables

Relevant for: API testing only

If you choose to create a data source from a database, UFT lets you retrieve data from both OleDB and ODBC databases. A wizard guides you through connecting to a database and creating an SQL query statement. After you complete the wizard, the Data pane displays the database's data tables.

The connection builder assists you in creating a string for your connection. For details, see "Connection Builder Dialog Box" on page 1720.

Once you are connected to the database, you can use UFT's Query Designer to create a custom SQL statement, based on the tables and views in your database.

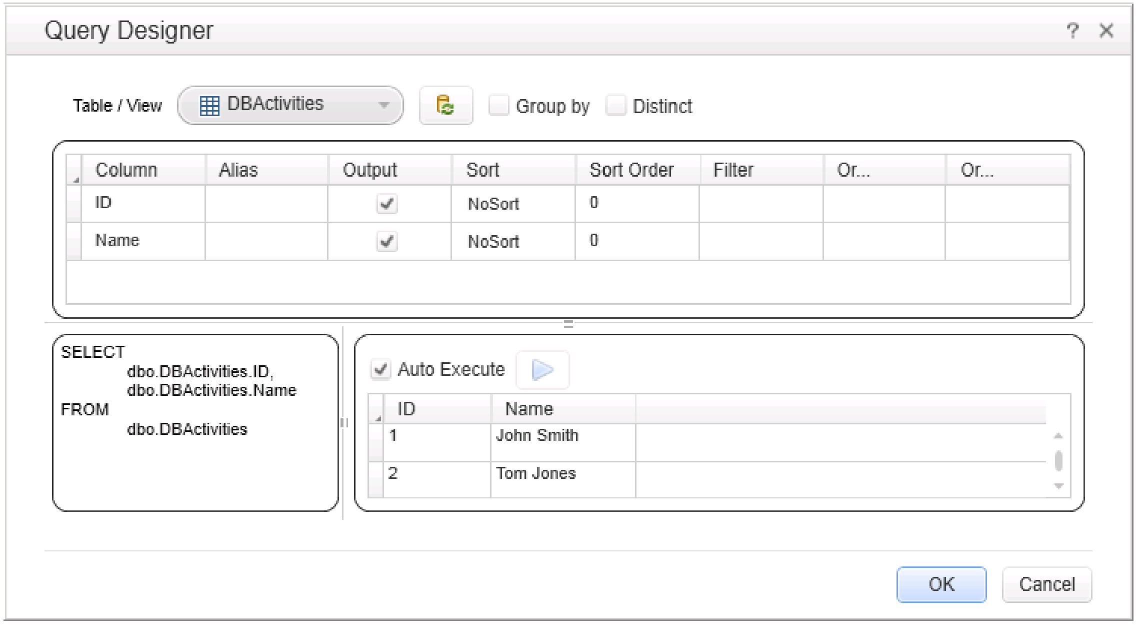


Table / View: DBActivities □ Group by □ Distinct

Column	Alias	Output	Sort	Sort Order	Filter	Or...	Or...
ID		✓	NoSort	0			
Name		✓	NoSort	0			

```
SELECT
        dbo.DBActivities.ID,
        dbo.DBActivities.Name
FROM
        dbo.DBActivities
```

☑ Auto Execute

ID	Name
1	John Smith
2	Tom Jones

For details on how to use the Query Designer, see the "Query Designer Dialog Box" on page 1724.

To learn more about using the wizard, see "Add a database data source" on page 235.

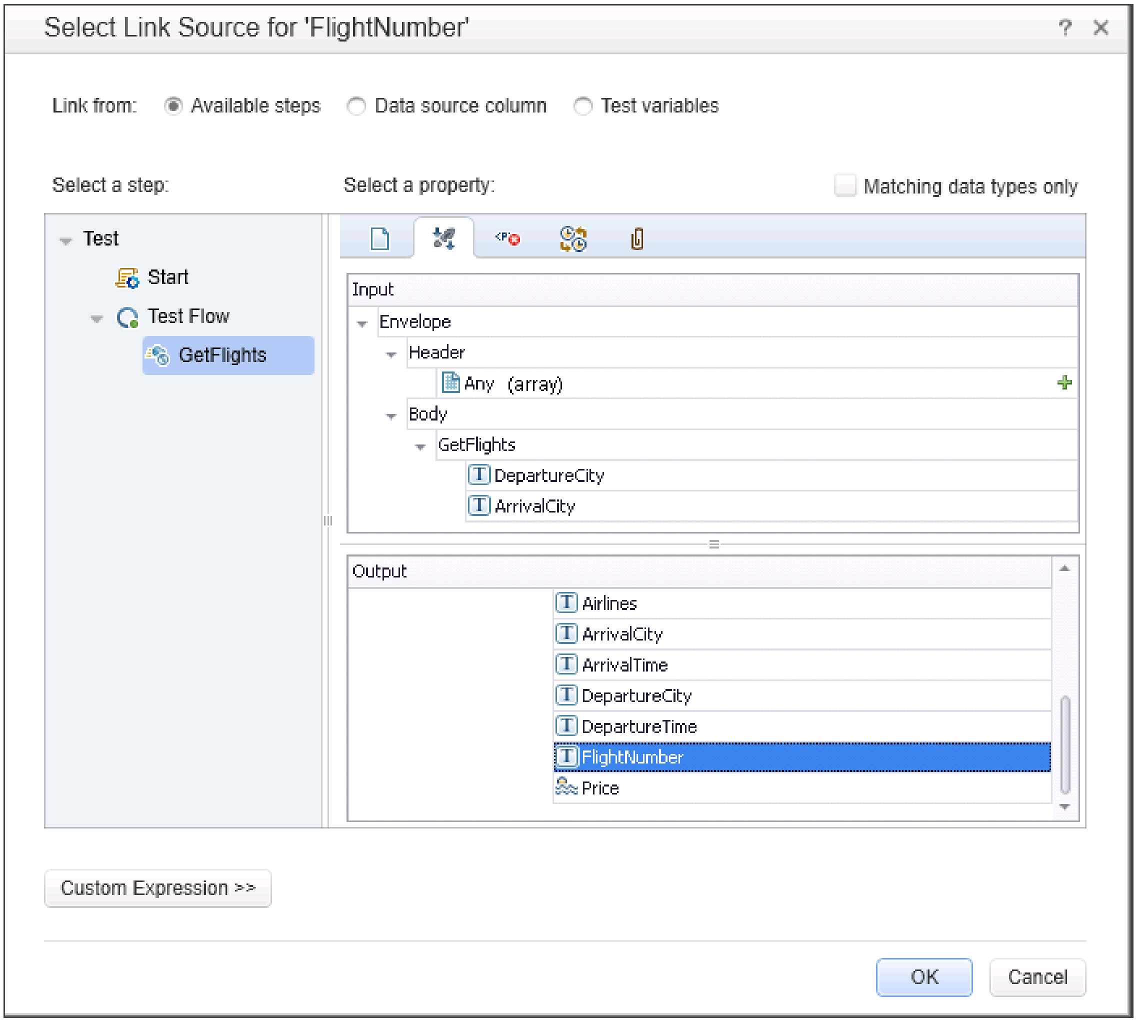
Linking Property Values to Other Test Steps

Relevant for: API testing only

In some instances, your application passes data from one application process to another. Thus, the input value that one process is calling is reliant upon the output value of another process.

Therefore, when testing the behavior and performance of your application in a test, you must be able to mimic this passing of data. UFT enables you to do this through linking the property values of one step to the property values of another step.

Linking step property values is much the same as linking a step's property values to a data source. You use the Select Link Source dialog box and select the option **Available Steps**.



Select Link Source for 'FlightNumber'

Link from: ● Available steps ○ Data source column ○ Test variables

Select a step:

- Test
  - Start
  - Test Flow
    - GetFlights

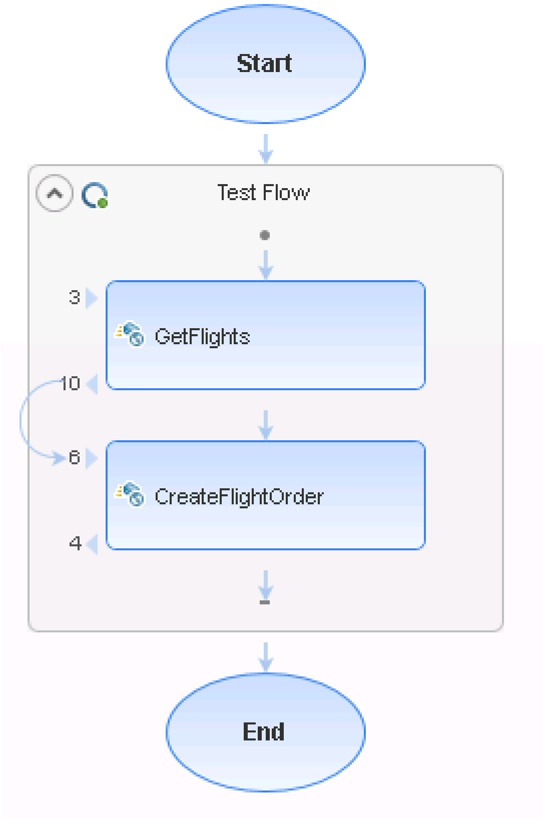
Select a property:

□ Matching data types only

Input
- Envelope
  - Header
    - Any (array)
  - Body
    - GetFlights
      - DepartureCity
      - ArrivalCity

Output
- Airlines
- ArrivalCity
- ArrivalTime
- DepartureCity
- DepartureTime
- FlightNumber
- Price

After you link the steps together, the canvas indicates the connection between the test steps:



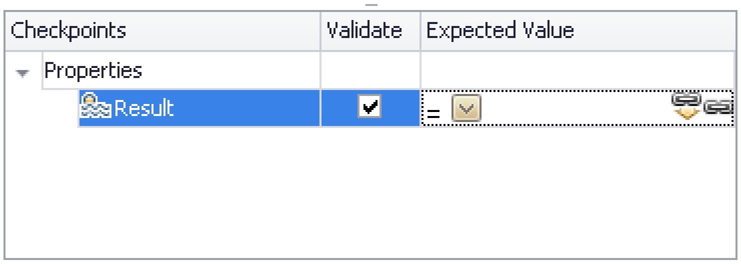
For task details, see "Link your test step to another step" on page 1820.

For details on the Select Link dialog box, see "Select Link Source Dialog Box (API Testing)" on page 1840.

Outgoing Links

Relevant for: API testing only

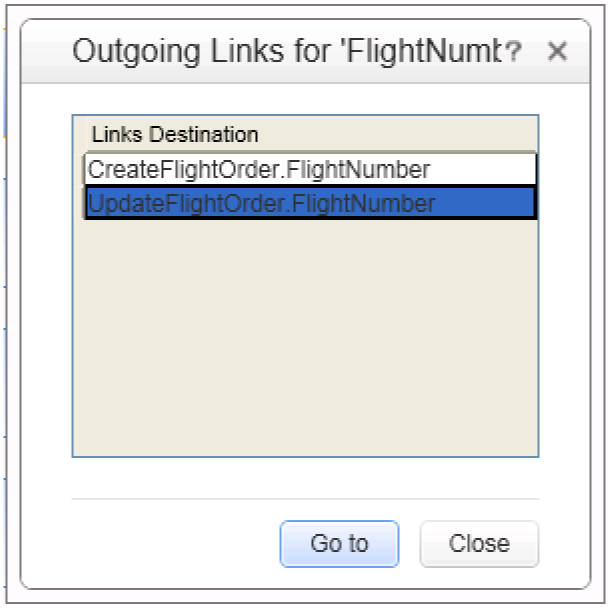
When you link a property to another step's property, UFT indicates this with a **Outgoing Links** button in the property's value column in the Properties pane's **Input/Checkpoints** tab.



Checkpoints	Validate	Expected Value
Properties		
Result	☑	=

By clicking on the button, you can view a list of the properties that link to this output property. This is especially helpful when linking other steps to an output property multiple times to enable you to view the flow of data throughout your test. Likewise, this helps you ensure that the test steps match the flow of properties/parameters as they do in your application.

When you click the **Outgoing Links** button, UFT opens a list of the target properties that link to this value. Using the **Go to** button, you can navigate directly to the linked property. This selects the target step on the canvas and the target property in the Properties pane.


Outgoing Links for 'FlightNumt?
Links Destination
CreateFlightOrder.FlightNumber
UpdateFlightOrder.FlightNumber
Go to
Close

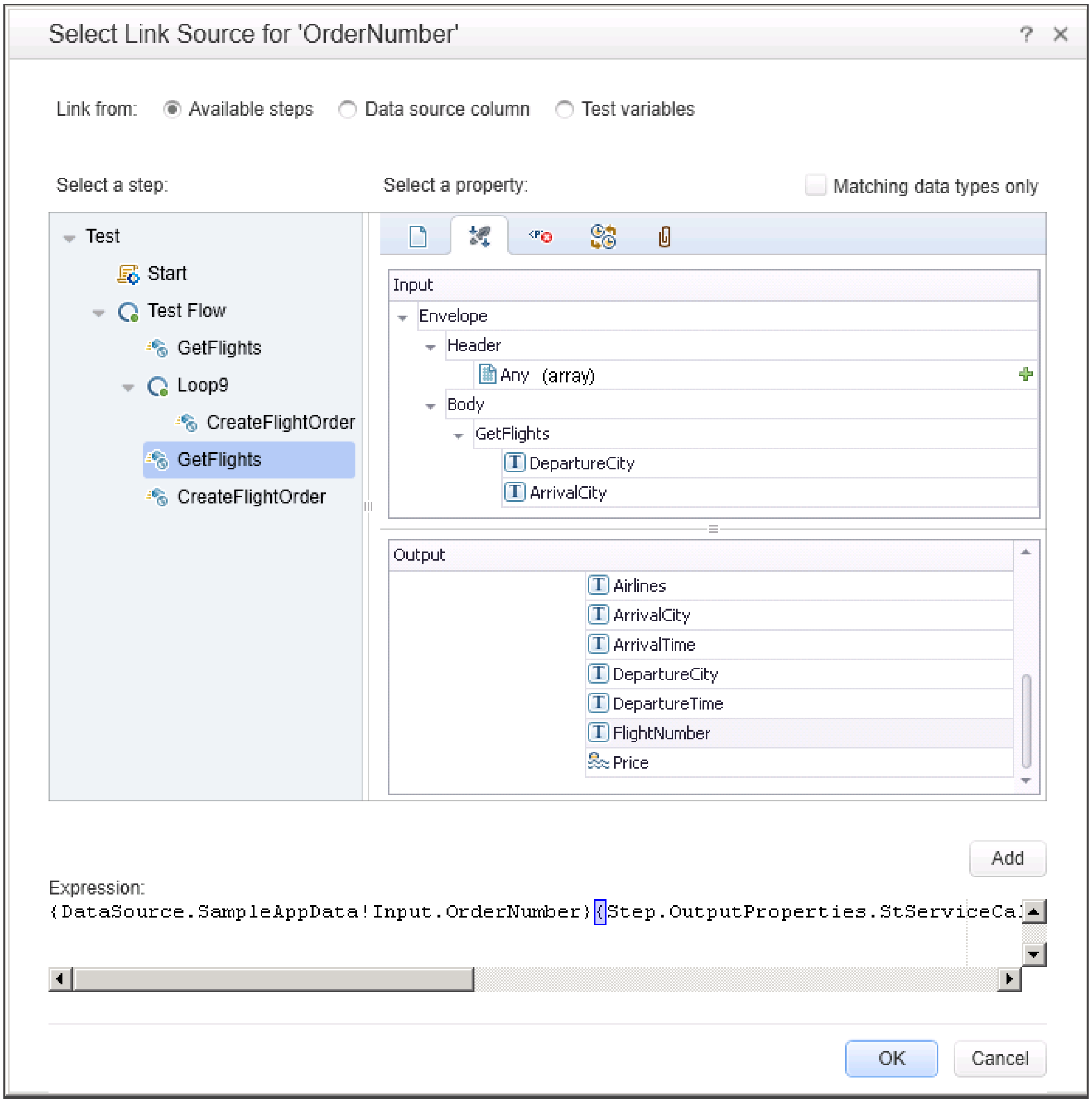
Linking Property Values to Multiple Sources of Data

Relevant for: API testing only

If the input of a particular application process in your application is linked to both a data source and the result of a previous application process, you can create property value expressions in UFT that reflect this process.

You do this by linking the property value of a test step to multiple sources of data and then creating custom expressions for the property value using a combination of the linking methods.

Using the Select Link Source dialog box, you create these custom expressions in any combination:



Select Link Source for 'OrderNumber'

Link from: ● Available steps ○ Data source column ○ Test variables

Select a step:

- Test
  - Start
  - Test Flow
    - GetFlights
    - Loop9
      - CreateFlightOrder
    - GetFlights
    - CreateFlightOrder

Select a property: □ Matching data types only

Input
- Envelope
  - Header
    - Any (array)
  - Body
    - GetFlights
      - DepartureCity
      - ArrivalCity

Output
- Airlines
- ArrivalCity
- ArrivalTime
- DepartureCity
- DepartureTime
- FlightNumber
- Price

Expression:

```
{DataSource.SampleAppData!Input.OrderNumber}{Step.OutputProperties.StServiceCa
```

The custom expression is then added as the property value for your test step and runs accordingly when you run your test.

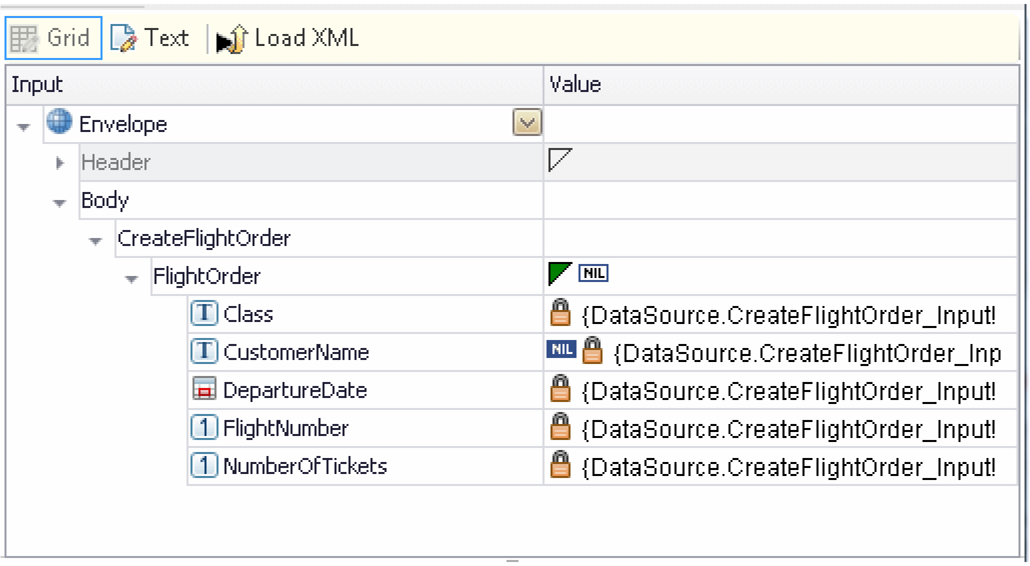
For task details, see "Link your test step to multiple sources" on page 1821.

For details on the Select Link dialog box, see "Select Link Source Dialog Box (API Testing)" on page 1840.

Data Driving

Relevant for: API testing only

Data-driving is the mechanism by which UFT automatically creates data expressions for properties. When you data drive a step, UFT adds data expressions to the step's value fields.



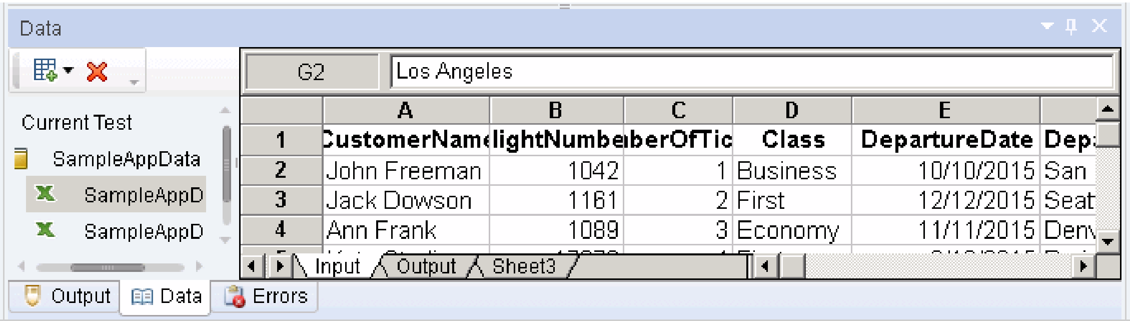
Input	Value
Envelope	
Header	
Body	
CreateFlightOrder	
FlightOrder	NIL
Class	{DataSource.CreateFlightOrder_Input!
CustomerName	NIL {DataSource.CreateFlightOrder_Inp
DepartureDate	{DataSource.CreateFlightOrder_Input!
FlightNumber	{DataSource.CreateFlightOrder_Input!
NumberOfTickets	{DataSource.CreateFlightOrder_Input!

These expressions refer to data in a new editable Excel table or XML structure, depending on the type of provider you specified during the data driving.

You can customize data-driving in the following ways:

- Specify the type of data source to create—Excel or XML.
- Indicate which properties to data drive:
  - For standard input and output (checkpoint) properties: **Input**, **Checkpoints**, or both.
  - For XSD files, loaded for the HTTP or SOAP Request steps: **Request Body**, **Response Body**, or both.
  - For additional types such as HTTP multipart and SOAP Fault properties, see the "Data Driving Dialog Box" on page 1834.
- Whether to use the same or two different data sources, when data driving two sections, such as Input and Checkpoint properties (Enabled only for Excel data sources).
- Whether or not to set the target step's loop, as a "For Each" type, based on the generated data source.

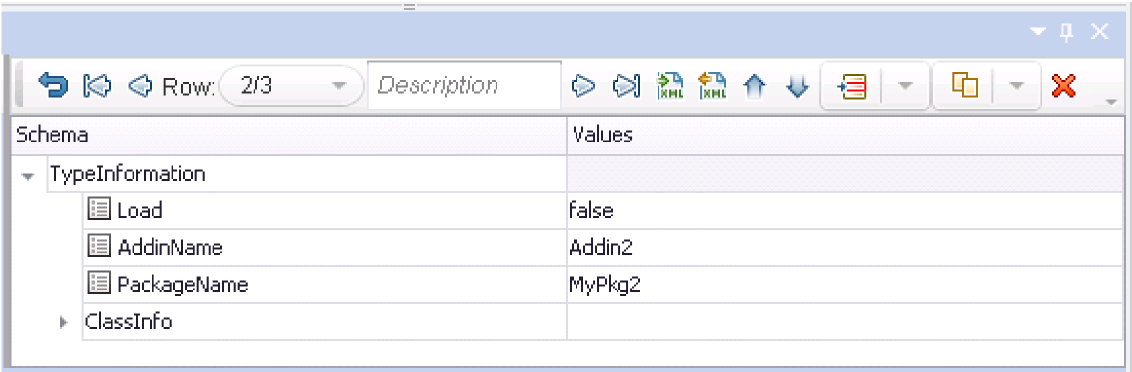
For Excel type tables, you can manually add to and edit the data in the Data pane.



G2 Los Angeles

	A	B	C	D	E	
1	CustomerNam	lightNumbe	berOfTic	Class	DepartureDate	Dep:
2	John Freeman	1042	1	Business	10/10/2015	San
3	Jack Dowson	1161	2	First	12/12/2015	Seat
4	Ann Frank	1089	3	Economy	11/11/2015	Denv

For XML data sources, you can manually edit the XML root.



Schema	Values
TypeInformation	
Load	false
AddinName	Addin2
PackageName	MyPkg2
ClassInfo	

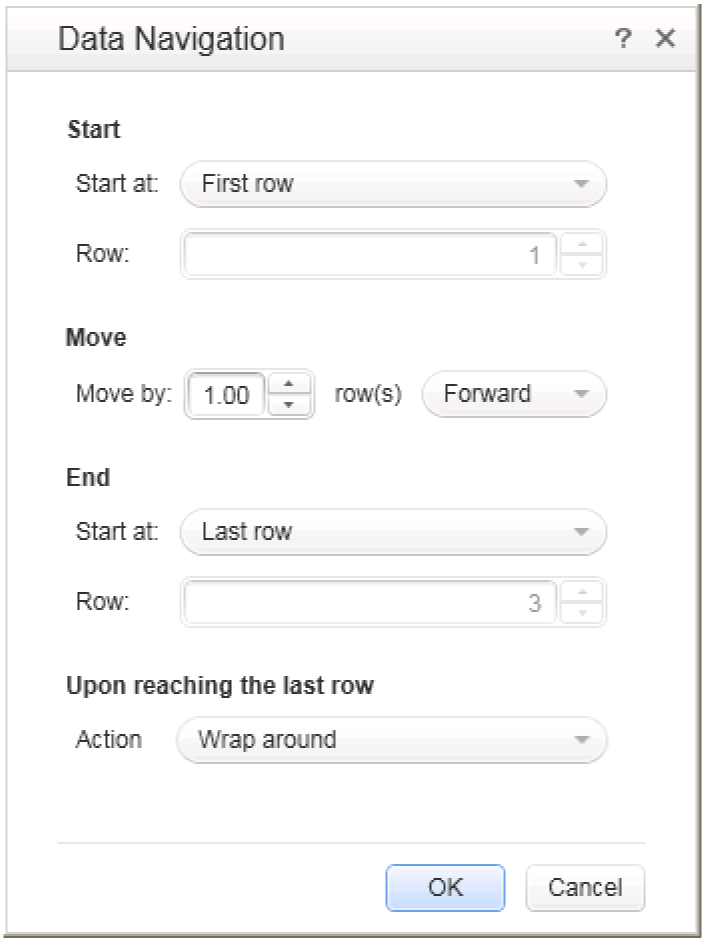
For task details, see "How to Data Drive an API Test Step" on page 1827.

For user interface details, see "Data Driving Dialog Box" on page 1834.

Navigating Within a Data Source

Relevant for: API testing only

Sometimes, when you are running your test, your data is stored in multiple sources and called by many different step properties in the test. In these cases, you can instruct UFT how to use the data sources, including where in the data source to begin calling data values, how to move through the data source, and where to stop using values from the data source. You specify these settings for each of the data sources associated with your test in the "Data Navigation Dialog Box" (described on page 1846)



Data Navigation

Start

Start at: First row

Row: 1

Move

Move by: 1.00 row(s) Forward

End

Start at: Last row

Row: 3

Upon reaching the last row

Action: Wrap around

This is useful to test your application's performance with constantly varying sets of data. In addition, this enables you to maintain your data source while still using it in a test. For example, if you are adding new data to a data source that is being used in a test, but the new data is not complete, you can instruct UFT to begin at the row containing the old data and stop before calling the new data values currently under construction.

The Data Navigation settings you create work with the loop settings for the Test Flow or selected test loop:

- If the loop is configured as a **ForEach** type, and you assign a specific data source as the loop's collection, then the Data Navigation settings affect the number of iterations of the loop. Because you specify the number of rows to use from the selected data source, the test runs the same number of iterations as the number of rows specified.

  For details on the different loop types, see "Flow Control Activities" on page 1651.

- For data sources that are not designated as the loop collection but whose values are called by steps within the loop, the Data Navigation policy affects the data differently. It indicates the order in which the values are called from the data source and assigned to steps within the loop. For example, if you specify in the Data Navigation dialog box to begin in the second row of your Excel data source, UFT populates the property value with the values from the second row (and so forth, as specified in the Data Navigation dialog box).

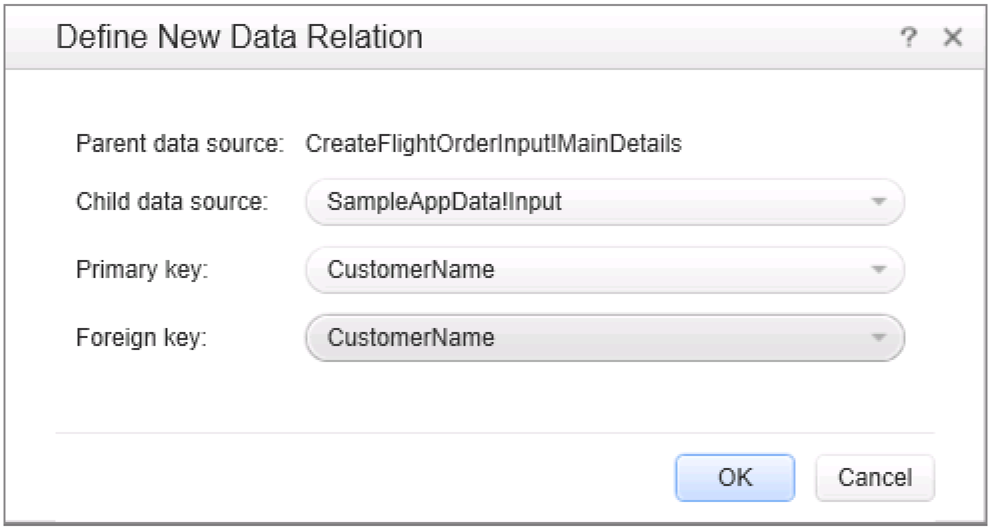
For task details, see "How to Set the Data Source Navigation Properties" on page 1825

Parent/Child Data Source Relations

Relevant for: API testing only

If your test uses multiple data sources to provide data for your test step properties, you can create a hierarchy of relations between data sources that describes how to use the data sources in your test. You use data relations when you use one specific data source as the primary data source for your test but require other data sources to populate property values within the same test.

When you use data relations, you assign the primary data source to the Test Flow or selected test loop. Then, using the Define New Data Relation dialog box, you specify any necessary child data sources and map the columns of the parent data source to the columns of the child data source.



Parent data source: CreateFlightOrderInput!MainDetails
Child data source: SampleAppData!Input
Primary key: CustomerName
Foreign key: CustomerName

When the test runs, UFT substitutes the child data as specified.

For task details, see "Create a new child relation" on page 1826.

For details on the Define New Data Relation dialog box, see the "Define New/Edit Data Relation Dialog Box" on page 1849.

Data Assignment in Arrays

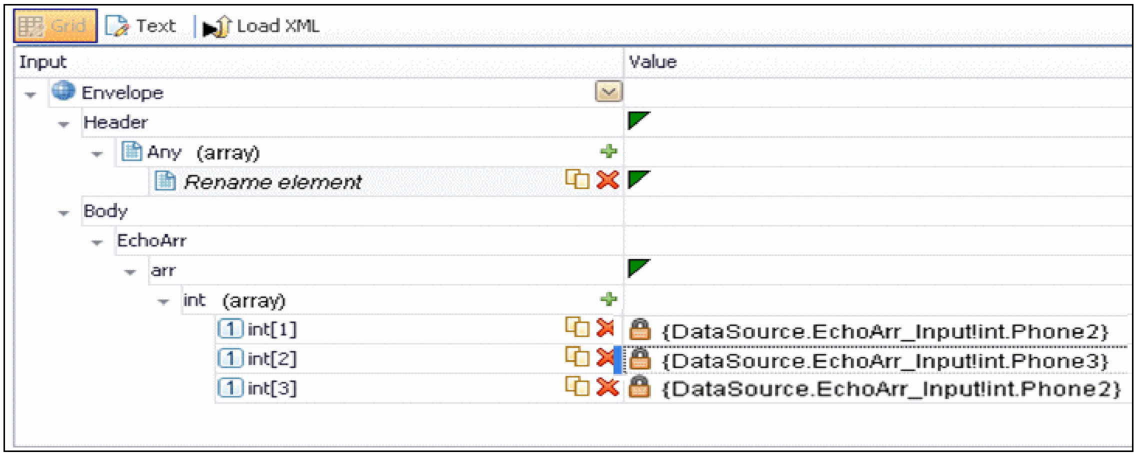
Relevant for: API testing only

When your step has properties that are arrays, you can assign them data as a fixed-size array or through data relations:

Fixed-Size Array Assignment

In the fixed-size method, you assign each element of a fixed-size array to any column in a data table. You can assign each array element to a different data table column.

The following example shows three array elements assigned to different columns of a data table.



Input	Value
Envelope	
Header	
Any (array)	
Rename element	
Body	
EchoArr	
arr	
int (array)	
int[1]	{DataSource.EchoArr_Input!int.Phone2}
int[2]	{DataSource.EchoArr_Input!int.Phone3}
int[3]	{DataSource.EchoArr_Input!int.Phone2}

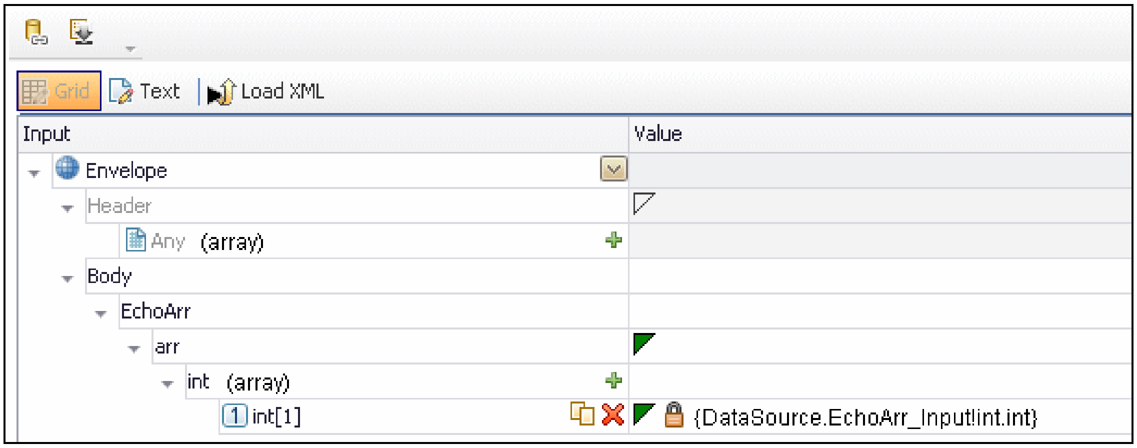
Data Relation Based Assignment

When you have one or more data relations defined, as described in "Create a new child relation" on page 1826, UFT assigns data from a single column. You link the first element of the array to a column in a data source.

In order for UFT to assign data based on a data relation, the following conditions must be present:

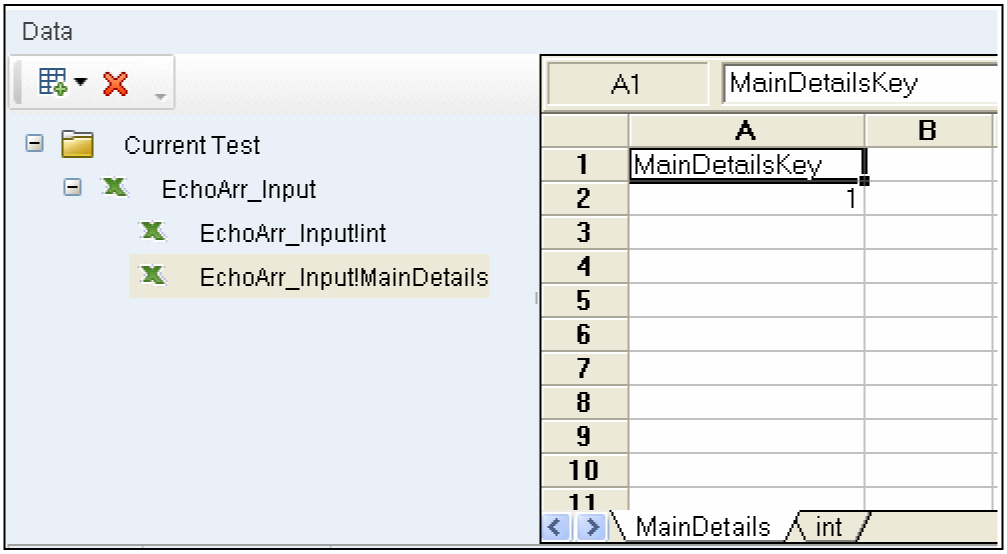
- The data source in the link is defined in a data relation.
- The parent data source is attached to the loop containing the step.
- Only the first element of the array is mapped to the child data source. If you map another array element to a different column, simple based assignment will be used.

The following example shows a single array element linked to a child data source. In this example, all elements of the array will take values from the same column, using the data-relation based assignment.



Input	Value
Envelope	
Header	
Any (array)	
Body	
EchoArr	
arr	
int (array)	
int[1]	{DataSource.EchoArr_Input!int.int}

The link to the first element indicates the column from which the values for all of the array elements will be taken.



A1 MainDetailsKey

	A	B
1	MainDetailsKey	
2	1	
3		
4		
5		
6		
7		
8		
9		
10		
11		

If you create a simple link to data, and the data source is already designated as a child in a data relation, the behavior will be relation-based.

Data-driven array elements will always be assigned using the data relation based assignment, using the column assigned to the first element.

Assignment Types

For relation-based data assignment, the drop down list adjacent to the name of the parent array node provides the following options for checkpoint validation:

- **Fixed**. Checks that each of the returned array elements matches the corresponding array element in the data table in the Data Pane. Each array is marked by an index number, as it checks the arrays by their index.

- **All**. Checks that all of the returned array elements match the array element in the Data Pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked >= 2 and the same property in another array element is set to <=10, the test run will check that all returned values are between 2 and 10.

- **Contains**. Checks that at least one of the returned array elements matches the value of the property in the Checkpoints pane. In this mode, arrays are not marked by an index number.

- **Fixed**. Checks that each of the returned array elements matches the corresponding array element in the data table in the Data Pane. Each array is marked by an index number, as it checks the arrays by their index.

- **All**. Checks that all of the returned array elements match the array element in the Data Pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked >= 2 and the same property in another array element is set to <=10, the test run will check that all returned values are between 2 and 10.

- **Contains**. Checks that at least one of the returned array elements matches the value of the property in the Checkpoints pane. In this mode, arrays are not marked by an index number.

Data Keywords

Relevant for: API testing only

You can use keywords to customize the test run and validation. For example, **SKIP** omits a property in the request or in the validation.

UFT provides keywords for both input properties and checkpoints. You can set keywords in the following ways:

- Select the property and choose **Insert Keyword** from the right-click menu.

- Link to a data source containing the keyword.

- Type to keyword into the **Expected Value** column (checkpoints only).

For the manual options, make sure to use the required format by enclosing the keyword with hash (#) signs. Keywords are not case-sensitive.

Input Keywords

The following keywords are supported for input properties:

Keyword	Description
#SKIP#	Omits this element from the XML of the request. This is useful for elements of SOAP requests, for which minOccurs = 0

Keyword	Description
#NIL#	Adds a **nil=true** attribute to the property's XML in the request. **Example:** <name John Doe nil="true"></name> If the XML element is not nillable, this is reported to the log.

Checkpoint Keywords

The following keywords are supported for checkpoints:

Keyword	Description
#EXISTS#	Verifies that the element is present in the XML response. In this evaluation, the actual value is ignored—it only checks for the presence of a value. This is useful for SOAP response elements, for which minOccurs = 0.
#NOT_ FOUND#	Verifies that the element is not present in the XML response. In this evaluation, the value is ignored. This is useful for SOAP response elements, for which minOccurs = 0.
#SKIP#	Informs the run engine to ignore this value when evaluating checkpoints.

To check if an element has a **nil="true"** attribute in its response, select or clear its NIL icon in the user interface.

Tasks

How to Assign Data to API Test/Component Steps

Relevant for: API testing only

This task describes the different ways to link test step properties (both for input properties and checkpoint properties) to data sources :

- "Manually enter the data for your test step properties" below
- "Link your test step to a data source" below
- "Link your test step to another step" on the next page
- "Link your test step to multiple sources" on page 1821
- "Link your test steps to a test or user-defined variable" on page 1822
- "How to Assign Data to API Test/Component Steps" above

Manually enter the data for your test step properties

1. In the canvas, select the test step for which you want to assign property values.
2. In the Properties pane, open the **Input/Checkpoints** tab .
3. In the **Value** cell for the property, enter the value.

> **Note:** If you are working with a step that uses XML or JSON input, you can also enter the property values using the **Text** view in the Input section. Click the Revert button to undo your changes in the XML view.

> **Tips:**
>
> - If the property value is a string, enter it exactly as you expect it appear (including spaces and special characters.)
> - If you want to use the values from the previous test run, click the **Load from Replay** button.

Link your test step to a data source

1. In the Data pane, associate a data source with your test. For details, see "How to Add Data Sources to an API Test" on page 233.

2. In the canvas, select the test step for which you want to assign property values.

3. In the Properties pane, open the **Input/Checkpoints** tab.

4. In the **Value** cell for the property, click the **Link to data source** button. The "Select Link Source Dialog Box (API Testing)" (described on page 1840) opens.

5. In the Select Link Source dialog box, select the **Data source column** option. A list of all data sources is displayed.

6. Select the data source containing your property's data value. A list of all available data columns is displayed.

7. In the data columns list, select the corresponding column name for your test step property and click **OK**.

   The property name in the Input/Checkpoints tab is now displayed with the associated data source name and data source value.

   > **Note:**
   >
   > - You can only link properties to data sources only if the data source is attached in a loop, such as **Test Flow** or custom loops.
   > - Linking a leaf node to a complex XML node is supported only for string type data.

Link your test step to another step

1. In the canvas, select the step for which you want to link a property value.

2. In the Properties pane, open the **Input/Checkpoints** tab.

3. In the **Value** cell for the property, click the **Link to data source** button. The "Select Link Source Dialog Box (API Testing)" (described on page 1840) opens.

4. In the Select Link Source dialog box, select the **Available steps** option. A list of all steps preceding the selected step is displayed.

5. In the list of available steps, select the step you want to link to the selected property value. A list of available properties is displayed in the right pane.

6. In the right pane, open the tab containing the property to which to link.

   > **Note:** If you are linking to the output of a previous step, open the **Input/Checkpoints** tab.

7. Select the property to which to link and click **OK**.

The property value is displayed in the **Value** column of the input property name with an expression representing the output of the linked step. The canvas also displays an arrow connecting he step property values.

> **Note:** The Properties pane displays a down arrow next to any property that is set to be used as output data for another step. Click the icon to display a list of all steps linked to the selected property. For details, see "Outgoing Links" on page 1808.

Link your test step to multiple sources

1. If necessary, add a data source. For details on adding data sources, see "How to Add Data Sources to an API Test" on page 233.

2. In the canvas, select the step for which you want to link a property value.

3. In the Properties pane, open the **Input/Checkpoints** tab .

4. In the **Value** cell for the property, click the **Link to data source** button . The "Select Link Source Dialog Box (API Testing)" (described on page 1840) opens.

5. In the Select Link Source dialog box, click the **Custom Expression** button to expand the expression area.

6. Create your expression, with any combination of methods. Click **Add** after linking to each source to add the value to your custom expression:

   - Manually enter the expression

   - Link to data source values, as described in "Link your test step to a data source"

     > **Note:** You can make a custom expression that includes multiple links to data sources.

   - Link to another step's property, as described in "Link your test step to another step"

     > **Caution:** If you are entering a custom string, make sure to add the string exactly as you want it to appear, including spaces and special characters.

   You can use these methods in any order and in any manner as needed to create your expression.

7. When you are finished entering all values, click **OK**.

   The custom expression is displayed in the **Value** column for the property name in the **Input/Checkpoints** tab exactly as you entered it in the expression area.

Link your test steps to a test or user-defined variable

1. In the canvas, select the step you want to link to a test or user-defined variable.
2. In the Properties pane, open the **Input/Checkpoints** tab .
3. In the **Value** cell for the property you want to link to a variable value, click the **Link to data source** button . The "Select Link Source Dialog Box (API Testing)" (described on page 1840) opens.
4. In the Select Link Source dialog box, select the **Test variables** option. The list of available variables is displayed.
5. In the list of available variables, select the variable for the link and click **OK**.

   The value for the selected property is displayed as the expression of the test variable.

How to Assign Data to API Test Steps - Tutorial

Relevant for: API testing only

This tutorial teaches you how to assign data to your test steps. The test steps are based upon the sample Flight API application provided with UFT.

> **Note:** For the task related to this scenario, see "How to Assign Data to API Test/Component Steps" on page 1819.

This tutorial includes the following steps:

- "Prerequisite - import the Web Service methods for the sample application" below
- "Associate a data source with your test" on the next page
- "Create your test steps" on the next page
- "Manually enter the input properties for the GetFlights step" on the next page
- "Link the CreateFlightOrder input properties to the data source" on page 1824
- "Link the Report step input properties to the CreateFlightOrder step" on page 1824
- "View the run results" on page 1825

1. Prerequisite - import the Web Service methods for the sample application

For this scenario, you use the **GetFlights** and **CreateFlights** methods from the sample Flight API application.

To import the service, do the following:

a. Open the Flight API application from the **Start** menu.

   For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

b. Type help to display the service details.

c. Copy the URL for the Web Service of the Flight API Application.

d. Import the Web Service into UFT. For details, see "How to Import a WSDL-Based Web Service" on page 1744.

2. Associate a data source with your test

Add the sample application Excel data source to your test. This file (SampleAppData.xlsx) is found in the **<UFT installation>\samples\flight_service** directory.

a. In the Data pane, click the **New Data Source** button and select **Excel**.

b. In the "New/Change Excel Data Source Dialog Box" (described on page 255), navigate to sample application Excel file. Click **OK** to add the Excel file to the test.

   The data source is displayed in the **Data** pane.

3. Create your test steps

From the Toolbox pane, drag the following steps to the canvas in order:

- **GetFlights** (found in the **Local Activities > Web Services** node)
- **CreateFlightOrder** (found in the **Local Activities > Web Services** node)
- **Report Message** (found in the **Miscellaneous** node)

4. Manually enter the input properties for the GetFlights step

To provide property values for the **GetFlights** step, use the first method for providing data by manually entering the property values.

To provide the property values, do the following:

a. In the canvas, make sure that the **GetFlights** step is selected.

b. In the Properties pane, open the **Input/Checkpoints** tab .

c. In the Input/Checkpoints tab, manually select the following values from the property value drop-down lists:

   - **DepartureCity:** Denver
   - **Arrival City:** Los Angeles

5. Link the CreateFlightOrder input properties to the data source

When the **GetFlights** method runs in the Flight API application, it automatically creates a number of output properties, including the airline number, price, a flight number, and so on.

In this step, you link the input property values of the **CreateFlightOrder** step both to the output of the **GetFlights** step and to the sample Excel file you associated with your test.

a. In the canvas, select the **CreateFlightOrder** step.

b. In the Properties pane, open the **Input/Checkpoints** tab .

c. In the Input/Checkpoints tab, in the **Value** cell for the **FlightNumber** input property, select the **Link to data** source button .

d. In the "Select Link Source Dialog Box (API Testing)" (described on page 1840), link the **FlightNumber** property to the **GetFlights** step output property of the same name. When prompted if you would like to link the selected property as part of a loop, select **No**.

   For details on linking to another test step property, see "Link your test step to another step" on page 1820.

   > **Note:** By default, the **FlightNumber** property is not visible. Expand the **GetFlightsResult** node and click the **Add** button to expand all the output properties.

e. In the Input/Checkpoints tab, link the remaining input properties to the relevant columns in the Excel data source.

   > **Note:** Make sure to clear the NIL property on any property values.

6. Link the Report step input properties to the CreateFlightOrder step

For the final step of this tutorial, you will create a custom expression to mimic the Flight API application's passing of flight data to a flight booking Web site page. For this, create the custom message that the results display.

a. In the canvas, select the **Report Message** step.

b. In the Properties pane, open the **Input/Checkpoints** tab .

c. In the Input/Checkpoints tab, in the **Value** cell of the **Message** property, select the **Link to data source** button 🔗.

d. In the "Select Link Source Dialog Box (API Testing)", click the **Custom Expression** button to display the expression area.

e. In the expression area, enter the text "Your flight order number is (with an extra space at the end).

f. In the "Select Link Source Dialog Box (API Testing)", select the Available step option.

g. In the list of available steps (left pane), select the **CreateFlightOrder** step.

h. In CreateFlightOrder properties list in the right pane, in the **Output Properties** section, expand the **CreateFlightOrderResult** node.

i. In the output properties list, select the **OrderNumber** property.

j. Click **Add** to add this to the custom expression.

k. Click **OK** in the Select Link Source dialog box to add this expression.

l. The expression Your flight order number is {Step.OutputProperties.StServiceCallActivity4.Body.CreateFlightOrderResponse.CreateFlightOrderResult.OrderNumber} appears in the Value column of the Message property

7. View the run results

Run the test. When the Run Results Viewer opens, expand the nodes in the Run Results tree and check the results of your steps in the Captured Data pane.

In the Response section of the Captured Data pane for the CreateFlight step and in the result summary for the Report Message step, you should see the data values passed between steps.

How to Set the Data Source Navigation Properties

Relevant for: API testing only

This task describes how to set the navigation preferences for data stored in tables. You can set different navigation properties for each data source. For details, see "Navigating Within a Data Source" on page 1813.

This task includes the following steps:

- "Add a data source to the Test Flow or test loop" on the next page
- "Specify the navigation properties for the data source" on the next page
- "Create a new child relation" on the next page

Add a data source to the Test Flow or test loop

1. Associate the necessary data sources with your test. For details, see "How to Add Data Sources to an API Test" on page 233.

2. In the canvas, select the **Test Flow** or another test loop.

3. In the Properties pane, select the **Data Sources** tab . The list of all currently associated data sources is displayed.

4. In the Data Sources tab, click **Add**. The "Attach Data Source to Loop Dialog Box " (described on page 1839) opens.

5. In the Attach Data Source to Loop dialog box, select the data source to add to the loop and click **OK**.

   The selected data source is now added to the loop and you can use it to link property values.

Specify the navigation properties for the data source

1. In the canvas, select the **Test Flow** or other test loop.

2. In the Properties pane, open the **Data Sources** tab . The list of data sources associated with the current loop is displayed.

3. In the Data Sources tab, click **Edit**. The "Data Navigation Dialog Box" (described on page 1846) opens.

4. In the Data Navigation dialog box, specify the **Start** and **End** rows.

5. Indicate the direction in which to move when retrieving data from the table, **Forward** or **Backward**, and the number of rows by which to advance for each iteration. Alternatively, you can indicate to move to a random row.

6. Specify the action to do when reaching the last row- **Wrap around** or **Keep using that row**. Click **OK**.

   For details on the Data Navigation dialog box, see "Data Navigation Dialog Box" on page 1846.

Create a new child relation

This step enables you to specify parent-child data source relations between multiple data sources. You can use parent-child relations for **Excel**, **Local Table**, and **Database** type data sources.

1. Add at least two or more Excel, Local Table, or Database data sources.

   > **Note:** This can also be a single Excel file with two worksheets.

2. In the Data pane, select the data source that you want to designate as a parent.

3. In the Properties, open the "Data Source Properties Tab (Properties Pane - API Testing)" (described on page 406).

4. Click the **Add** button. The "Define New/Edit Data Relation Dialog Box" (described on page 1849) opens.

5. Specify the details of the new relation:

   - The data source to use as the child data source

   - The primary key - the data column in the parent data source

   - The foreign key - the data column in the child data source. This column is used in place of the primary key column in the parent data source when running the test.

   Click **OK** after specifying the data source details. The data relation is displayed in the Data Source Properties tab for the parent data source.

How to Data Drive an API Test Step

Relevant for: API testing only

This task describes how to data drive a test step. Data driving is UFT's mechanism for automatically creating data expressions for step properties.

For further details, see "Data Driving" on page 1811.

This task includes the following steps:

- "Prerequisite" below

- "Open the Input Properties tab" below

- "Include the data " below

- "Data drive the test step" on the next page

- "Enter the data source values" on the next page

1. ### Prerequisite

   Create a step in the canvas. For details, see "Add activities to the test to create test steps" on page 1599.

2. ### Open the Input Properties tab

   Display the step's properties grid. In the Properties pane,open the **Input/Checkpoints** tab .

3. ### Include the data

   To include optional properties in the data driving, toggle the triangle icon adjacent to the property. A filled-in triangle indicates an included property.

4. **Data drive the test step**

   a. In the Input/Checkpoints tab, click the **Data Drive** button. The "Data Driving Dialog Box" (described on page 1834) opens.

   b. In the Data Driving dialog box, choose a Data provider: **Excel** or **XML.**

   c. Select the properties upon which you want to apply data driving: **input properties**, **checkpoints**, or both.

   d. If you specified an **Excel** provider and data-driving for both input properties and checkpoints, specify whether to place the data for both sections in the same Excel worksheet or different ones.

   e. Indicate whether you want to **Configure '<loop name>' as a For Each loop using the new data source**. This option repeats the steps in the loop frame according to the number of data rows in the Excel or XML data source. If you disable this option, you can manually set the number of iterations via the loop properties. This setting overrides the general loop properties. For details, see "Loop Activity" on page 1652.

   f. Click **OK**.

   g. UFT prompts you to confirm the action. Click **OK**.

      UFT populates the value column with data expressions. The expressions are preceded by informational icons, indicating read-only status or unmatching data types.

      > **Note:** If your properties are within an array, you must create at least one array element to allow data driving. To add an array element, select the array's parent node and click the **Add** button + . If you do not add at least one element, then the array will not be data driven.

      For user interface details, see the "Data Driving Dialog Box" on page 1834.

5. **Enter the data source values**

   In the Data pane, edit the contents of the newly created data tables or XML source.

How to Data Drive Array Checkpoints

Relevant for: API testing only

This task describes how to set checkpoints for elements of an array through data driving. For details about setting regular checkpoints, see "How to Run an API Test" on page 643.

This task includes the following steps:

- "Enable active content on your computer" below
- "Add a step with an array output" below
- "Add an array element" below
- "Data Drive the array" below
- "Set the evaluation expression" below
- "Provide data for the array" on the next page
- "Select an array validation method" on the next page
- "Set the number of iterations - optional" on the next page
- "Run the test" on the next page
- "Open the Checkpoint report" on the next page

1. ## Enable active content on your computer

   The Run Results Viewer shows array checkpoint results in an expandable tree. To enable this view, configure your browser to allow active content as described in "How to Set Array Checkpoints for Test Steps" on page 1615.

2. ## Add a step with an array output

   Add a test step with output properties in the form of an array.

3. ## Add an array element

   a. Open the Properties pane and click in the **Checkpoints** area.

   b. Use the plus button ✚ . in the row of the parent node, to add an array element.

   c. Provide property values. These values are transferred to the Data pane during data driving. If you do not provide data, include the array element by selecting the triangle icon in the parent node.

4. ## Data Drive the array

   a. Select the array's parent node and click the **Data Drive** button .

   b. In the "Data Driving Dialog Box" (described on page 1834), select **Only Input, Checkpoints** or **Both Input and Checkpoints** as a **Data Driven Section** option.

   c. Click **OK**. The data driving mechanism informs you that it succeeded.

5. ## Set the evaluation expression

In the Properties pane, adjacent to the data driving expression, select an evaluation operator, such as **=**, **<**, and so forth.

6. **Provide data for the array**

   a. Once you data drive an array, you do not set property values from the **Checkpoints** pane. Instead, you edit the table in the Data pane. In the Data pane, which opens automatically, click the node in the left pane, corresponding to the array element that is to be validated.

   b. Enter the iteration number to which to checkpoint should be applied in the MainDetails column of the **<array_name>** and MainDetails tables. A "1" indicates the first iteration. If you have several columns with "1" as the iteration number, the checkpoints will be validated against all of those values.

7. **Select an array validation method**

   Select the step in the canvas and view the Properties pane. Expand the drop down adjacent to the name of the parent array node. Select one of the following:

   - **Fixed.** Checks that each of the returned array elements matches the corresponding array element in the data table in the Data pane. Each array is marked by an index number, as it checks the arrays by their index.

   - **All.** Checks that all of the returned array elements match the array element in the Data pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked >= 2 and the same property in another array element is set to <=10, the test run will check that all returned values are between 2 and 10.

   - **Contains.** Checks that at least one of the returned array elements matches the value of the property in the **Checkpoints** pane. In this mode, arrays are not marked by an index number.

8. **Set the number of iterations - optional**

   If you selected the **Configure Test Flow as a ForEach loop using the new data source option** in the "Data Driving Dialog Box" (described on page 1834), you can skip this step. If not, click in the **Test Flow** or **Loop** box and open the Properties pane. Set the test flow properties, such as the number of iterations or loop type.

9. **Run the test**

   Click the **Run** button and provide a results location. For details, see "How to Run an API Test" on page 643.

10. **Open the Checkpoint report**

    In the Run Results Viewer, expand the test results tree in the left pane. Select a Checkpoint node. In the **Captured Data** pane, click **View Report** in the Details column. In the report that opens, expand the checkpoint node for each of the iterations to view its actual and expected value.

How to Parameterize XML Data

Relevant for: API testing only

This task describes how to parameterize XML data. Suppose you pasted or loaded an XML file as the request body for your step. Using the parameterization capabilities, you can replace a constant value with a data source expression.

1. **Prerequisites**

   Create a test step that uses XML text in the request body. This applies to Network and XML type activities. If you intend to parameterize the data with a data source, import or create the data source. For details, see "How to Add Data Sources to an API Test" on page 233.

2. **Add the XML body text**

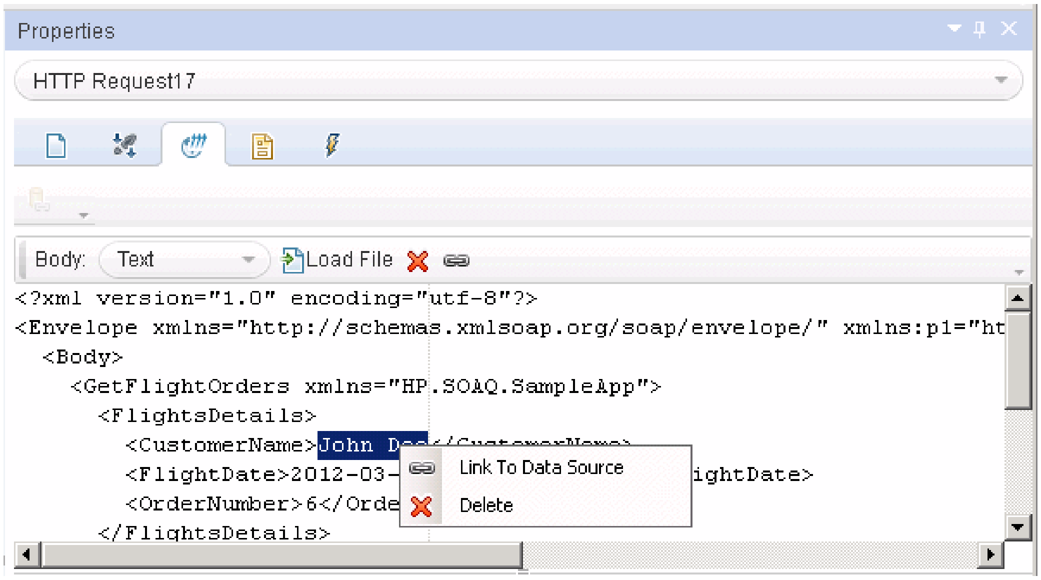
   a. Using the Properties pane, open the **HTTP** tab .

   b. Click the **Load File** button and load an XML file or paste XML code directly into the **Body** area.

3. **Select the text to parameterize**

   a. In the **HTTP** tab, select **Text** from the **Body** type drop-down list.

   b. In the Body area, select the text value that you want to replace and select **Link to Data Source** from the right-click menu.

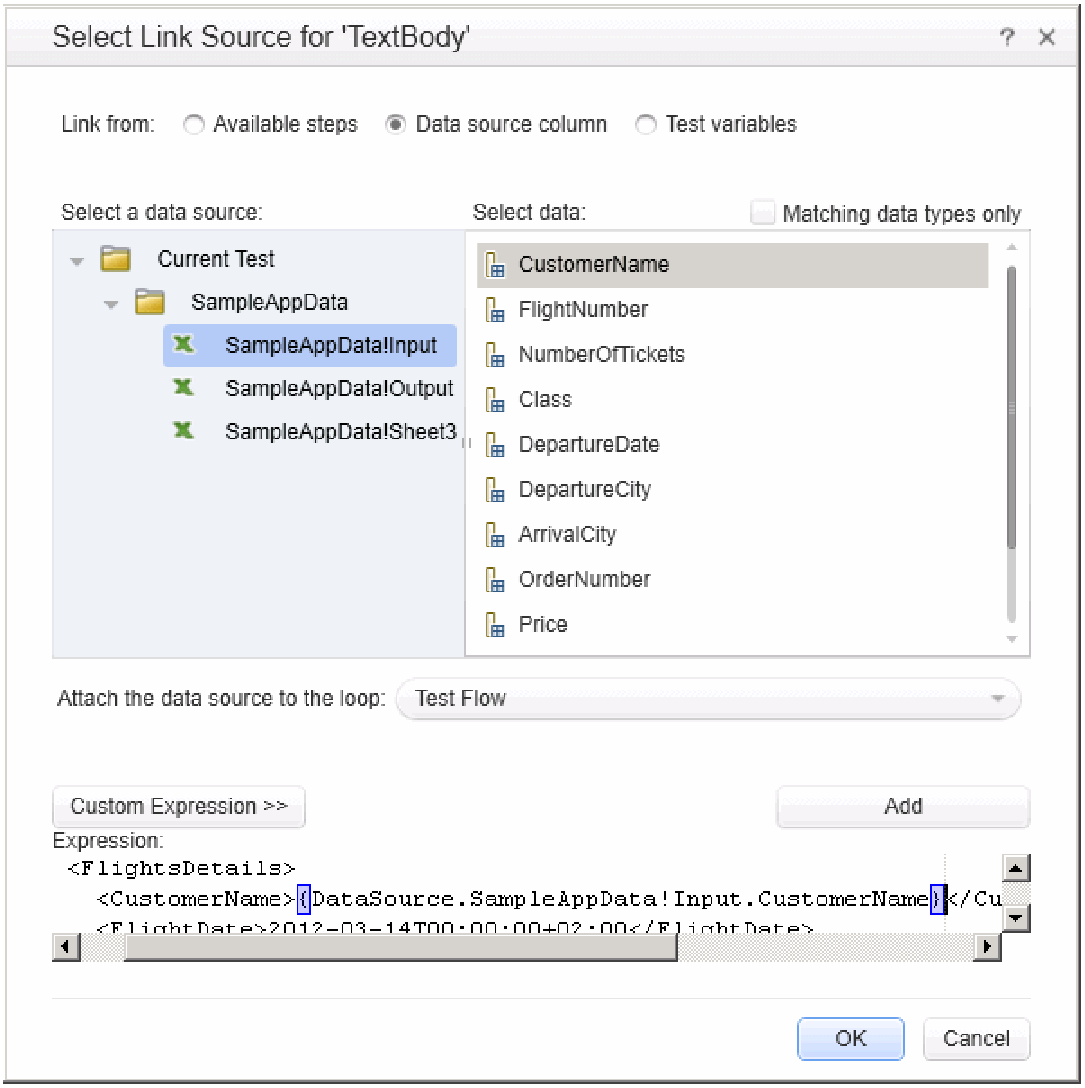


```
<?xml version="1.0" encoding="utf-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:p1="ht
  <Body>
    <GetFlightOrders xmlns="HP.SOAQ.SampleApp">
      <FlightsDetails>
        <CustomerName>John Doe</CustomerName>
        <FlightDate>2012-03-                        ightDate>
        <OrderNumber>6</Orde
      </FlightsDetails>
```

4. **Select a link source**

   In the Select Link Source dialog box, select a data source type, such as **Available steps** or **Data source column**. Select the node that you want to use for parameterization. Click **Add**.

Note that the expression changed.



Select Link Source for 'TextBody'

Link from: ○ Available steps ● Data source column ○ Test variables

Select a data source:

- Current Test
  - SampleAppData
    - SampleAppData!Input
    - SampleAppData!Output
    - SampleAppData!Sheet3

Select data: □ Matching data types only

- CustomerName
- FlightNumber
- NumberOfTickets
- Class
- DepartureDate
- DepartureCity
- ArrivalCity
- OrderNumber
- Price

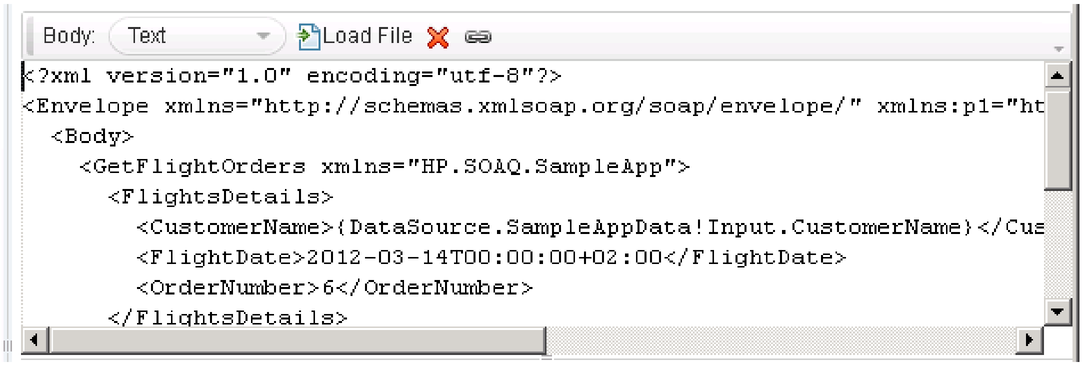
Attach the data source to the loop: Test Flow

Expression:

```
<FlightsDetails>
  <CustomerName>{DataSource.SampleAppData!Input.CustomerName}</Cu
  <FlightDate>2012-03-14T00:00:00+02:00</FlightDate>
```

5. **Verify the value in the Properties pane**

In the Properties pane, check the Body area, and verify that the constant value was replaced with the data source expression.



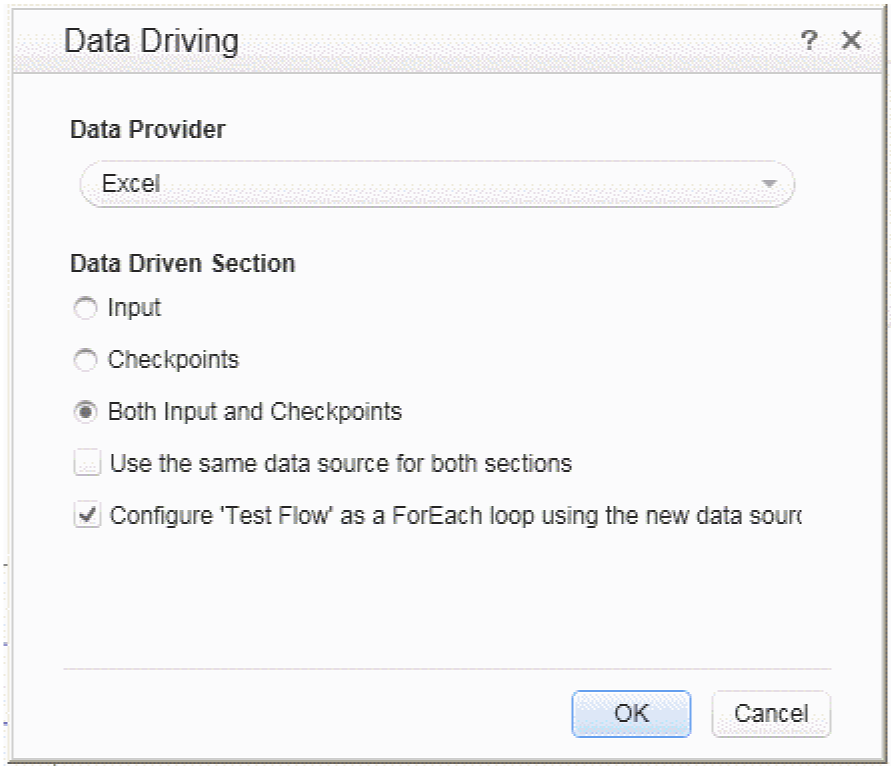
```
<?xml version="1.0" encoding="utf-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:p1="ht
  <Body>
    <GetFlightOrders xmlns="HP.SOAQ.SampleApp">
      <FlightsDetails>
        <CustomerName>{DataSource.SampleAppData!Input.CustomerName}</Cus
        <FlightDate>2012-03-14T00:00:00+02:00</FlightDate>
        <OrderNumber>6</OrderNumber>
      </FlightsDetails>
```

Reference

Data Driving Dialog Box

Relevant for: API testing only

Enables you to populate the input property and checkpoint values for a step, with data expressions.



Data Provider

Excel

Data Driven Section

○ Input

○ Checkpoints

● Both Input and Checkpoints

□ Use the same data source for both sections

☑ Configure 'Test Flow' as a ForEach loop using the new data sourc

To access	1. Select a step within the canvas. 2. In the Properties pane, open the **Input/Checkpoints** tab . 3. Click in the Properties pane's toolbar.
Relevant tasks	"How to Data Drive an API Test Step" on page 1827
See also	"New/Change Excel Data Source Dialog Box" on page 255

The following elements are included:

UI Elements	Description
Data Provider	The source type of the data: **Excel** or **XML**.

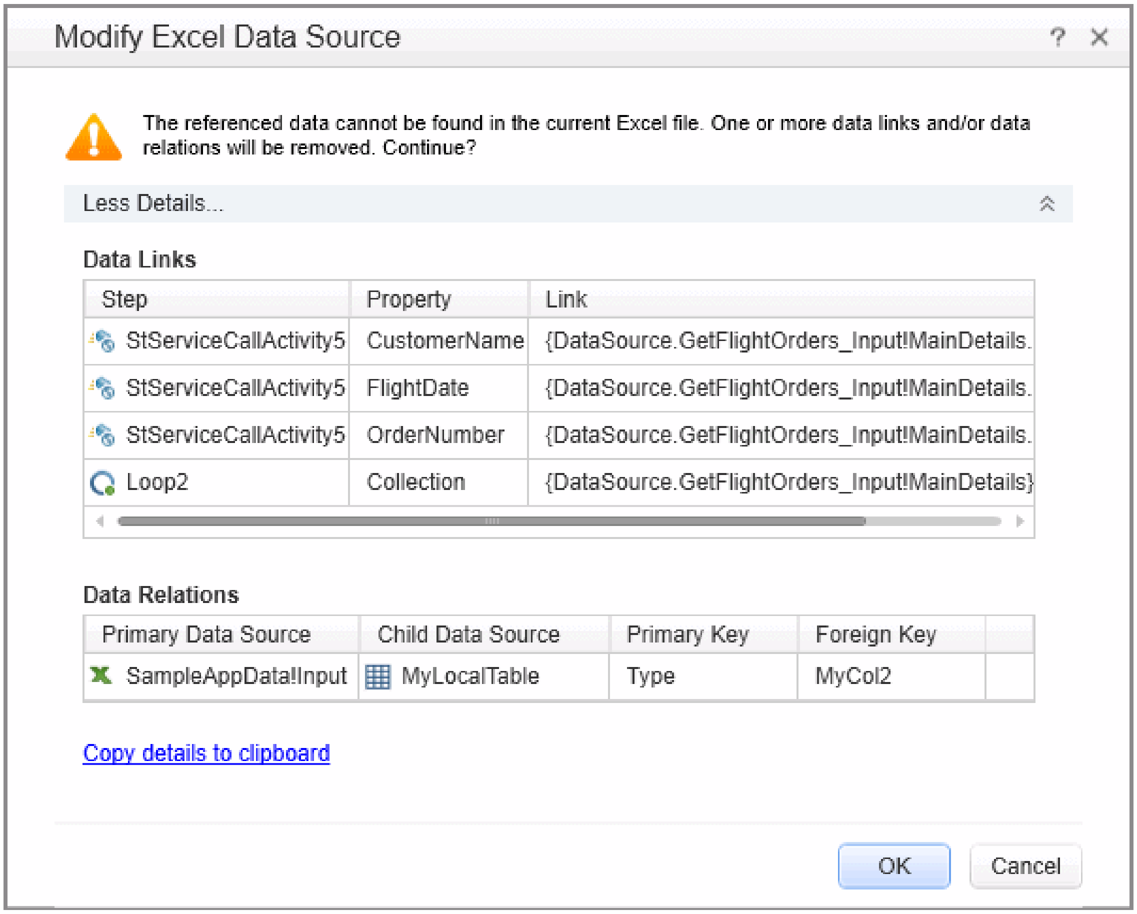
UI Elements	Description
Data Driven Section - for Input/Checkpoint properties	The type of properties upon which to apply the data driving. These options are available when data driving standard input and output checkpoints properties. • **Input**. All input properties listed in the grid. • **Checkpoints**. All checkpoints listed in the grid. For array type properties, you must create at least one array element in order to data drive the array. • **Both Input and Checkpoints**. All properties. **Note:** Activities that do not have output parameters, such as **Report Message** and **System** type activities, will only show the **Input** option.
Data Driven Section - for Request/Response	The schema to data drive. These options are available when data driving XSD schema files for an **HTTP**, **SOAP Request**, or **REST** step. • **Request Body**. All input properties associated with the request body. • **Response Body**. All output properties associated with the response body. For array type properties, you must create at least one array element in order to enable data driving. • **Both Request Body and Response Body**. All entities in both schemas. **Note:** These options are only available after importing a schema into the **Input** and/or **Checkpoints** sections.
Data Driven Section - for SOAP Fault properties	**Fault Properties.** The SOAP Fault properties to data drive. This option is available when invoking data driving in the Properties pane's **SOAP Fault** view. **Note:** These options are only available for Web Service and SOAP Request steps.

UI Elements	Description
Data Driven Section - for Multipart properties	The multipart properties to data drive. These options are available for an HTTP step with a multipart message. • **Request MultipartInfo**. Only Request multipart properties. • **Response MultipartInfo**. Only Response multipart properties. • **Both Request MultipartInfo and Response MultipartInfo**. All multipart properties.
Use the same data source for both sections	Uses the same data source for the Input/Checkpoint properties or Request/Response body. When selected, UFT creates a single Excel worksheet with columns for both the Input and Output properties. For example, for the sample application's **CreateFlightOrder** step, UFT creates a single worksheet with columns for the **Input** properties: Class, CustomerName, DepartureDate, FlightNumber, NumberofTickets, and the **Output** properties: OrderNumber, and TotalPrice. **Note:** This option is available only when selecting the **Both** option, for an Excel Data Provider type.
Configure 'Test Flow/Loop' as a ForEach loop using the new data source	Runs the steps in the Test Flow or Loop using a **For Each** type loop, with the values from the newly created data source. The number of iterations is determined by the number of rows in the data source—it performs one iteration for each row. This setting is applied to the innermost loop containing the data-driven step. **Note:** This option overrides the configuration that you may have set for the Test Flow/Loop settings. For details, see "Loop Activity" on page 1652.

Data Link/Relation Message Box

Relevant for: API testing only

This message box lists the elements that were not found in the new Excel file, when you changed the data source's existing Excel file.



Modify Excel Data Source

The referenced data cannot be found in the current Excel file. One or more data links and/or data relations will be removed. Continue?

Less Details...

Data Links

Step	Property	Link
StServiceCallActivity5	CustomerName	{DataSource.GetFlightOrders_Input!MainDetails.
StServiceCallActivity5	FlightDate	{DataSource.GetFlightOrders_Input!MainDetails.
StServiceCallActivity5	OrderNumber	{DataSource.GetFlightOrders_Input!MainDetails.
Loop2	Collection	{DataSource.GetFlightOrders_Input!MainDetails}

Data Relations

Primary Data Source	Child Data Source	Primary Key	Foreign Key
SampleAppData!Input	MyLocalTable	Type	MyCol2

Copy details to clipboard

To access	Do the following: 1. Save the test. 2. In the Data Pane, select the main node of an existing Excel data source. 3. In the Properties pane, select **Change Excel File**. 4. In the "New/Change Excel Data Source Dialog Box""New/Change Excel Data Source Dialog Box"(described on page 255), specify a new Excel file. 5. Click **OK** to permanently remove all of the listed items and save the test. **Note:** This dialog box only opens if the new Excel file has renamed or missing worksheets or columns, which affect data links or data relations.
Relevant tasks	• "Add an Excel data source" on page 234 • "How to Assign Data to API Test/Component Steps" on page 1819
See also	• "New/Change Excel Data Source Dialog Box" on page 255 • "Navigating Within a Data Source" on page 1813 • "Parent/Child Data Source Relations" on page 1814

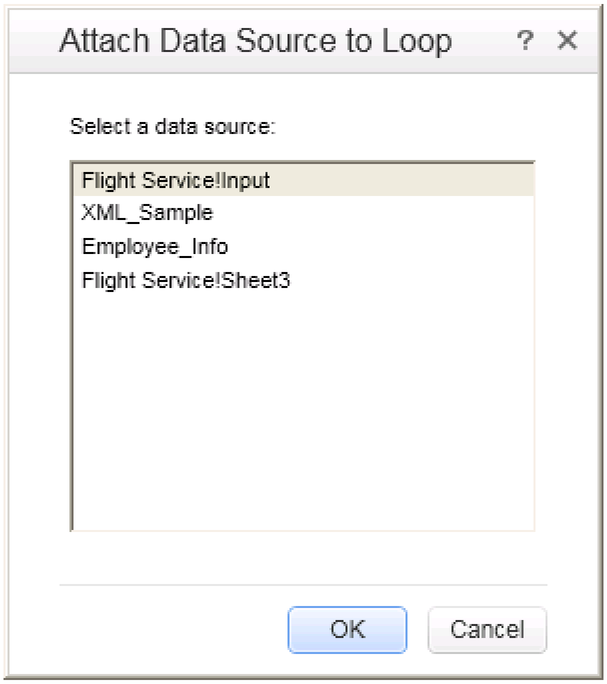
User interface elements are described below:

UI Elements	Description
More Details/Less Details	Expands or collapses the **Details** section to show or hide data links or data relations that will be removed when you click **OK**.
Data Links table	A table of all of the affected data links. The table displays the step name, the affected property of the step, and the specific data link.
Data Relations table	A table of all of the affected data relations. The table lists the name of the **Primary Data Source**, the name of the **Child Data Source**, the affected column in the primary data source (**Primary Key**) and the affected column in the child data source (**Foreign Key**).
Copy details to clipboard	Copies the details to the clipboard, in a comma separated format.

Attach Data Source to Loop Dialog Box

Relevant for: API testing only

This dialog box enables you to add a data source to the current loop.



To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. Associate a data source or multiple data sources with your test in the Data pane 3. In the canvas, select the **Test Flow** or another test loop. 4. In the Properties pane, open the **Data Sources** tab . 5. Click the **Add** button.

Important	- You must first define data in the Data pane before you generate a list of data sources. - When you link any step in the loop to the data, it automatically adds a data source to the loop. For details, see "How to Assign Data to API Test/Component Steps" on page 1819.
Relevant tasks	"How to Set the Data Source Navigation Properties" on page 1825
See also	"Data Navigation Dialog Box" on page 1846

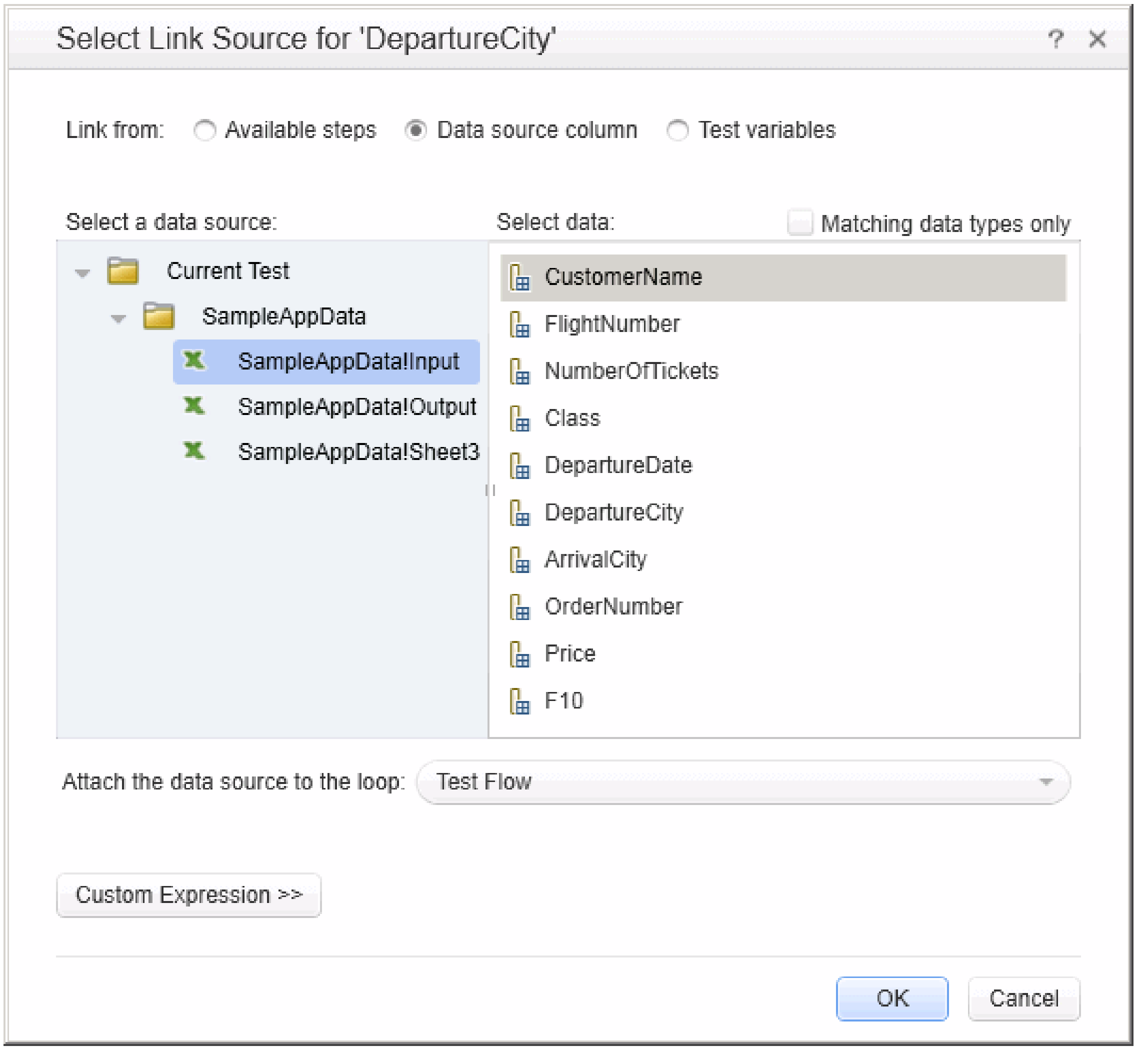
User interface elements are described below:

UI Element	Description
Select a data source	A list of the available data sources.

Select Link Source Dialog Box (API Testing)

Relevant for: API testing only

This dialog box enables you to select a data source for the step's properties.



Select Link Source for 'DepartureCity'

Link from: ○ Available steps ● Data source column ○ Test variables

Select a data source:

- Current Test
  - SampleAppData
    - SampleAppData!Input
    - SampleAppData!Output
    - SampleAppData!Sheet3

Select data: □ Matching data types only

- CustomerName
- FlightNumber
- NumberOfTickets
- Class
- DepartureDate
- DepartureCity
- ArrivalCity
- OrderNumber
- Price
- F10

Attach the data source to the loop: Test Flow

To access	Click the **Link to data source** button in the **Value** column of the property for which you want to select a data source.
Important information	• To see the icon, move the cursor to the right of the property value area. • A step property not contained within a loop, cannot be linked to a data source. For example, the For Loop's **Number of Iterations** property, cannot be linked to a data source, since it is not contained within the Test Flow loop.

Relevant tasks	"How to Assign Data to API Test/Component Steps" on page 1819
See also	- "How to Add Data Sources to an API Test" on page 233 - "How to Set the Data Source Navigation Properties" on page 1825

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

UI Elements	Relevant for	Description
Link from:	All types	The source for the property value: - **Available steps.** A property value from another step. - **Data source column.** A value from a data source—Excel, XML, local table, or database. - **Test variables.** A value of an test variable—either a user-defined variable or a system variable.
Select a step	Available steps	A tree hierarchy of the steps whose properties are available as data for the current step. This list could include: - Previous steps. - An input value of the current step, when selecting data for a property in the **Checkpoints** section. - For loop type steps, previous steps or other steps within loop.
Select a property	Available steps	The step's linkable properties. The displayed sections depend on the step type. For most step types, the Checkpoints section is displayed.
Select data	- Available steps - Data source column	- For table-based data sources such as Excel, local tables and database: a list of the columns in the data table. - For XML data sources: a hierarchy of the data source's nodes.
Matching data types only	Data source column	Hides all columns (or schema rows for XML data sources) whose data types do not match the properties of the step whose value you want to set.

UI Elements	Relevant for	Description
Attach the data source to the loop	Data source column	A drop down list of the loops in the test containing the step. The data source is only associated with the selected loop. **Default:** Innermost loop containing the step.
<user variable grid>	Test variables	A list of the user-defined test variables and their values in the current profile.

UI Elements	Relevant for	Description
<system variable grid>	Test variables	A list of the system environment variables and their values: • IsLoadTest • TestDir • TestName • LocalHostName • OS • OSVersion • ProductDir • ProductName • ProductVer • UserName • CurrentDate • QcUrl • QcUserName • QcPassword • QcDomain • QcProject The following variables are available when running the test on a remote agent: • CurrentRunID • CurrentTestSet • CurrentTestSetID • CurrentTestInstanceID
Custom Expression	All types	Expands the dialog box to allow you to manually edit the link and create a custom expression. This is useful for creating an expression that uses multiple sources.

Data Retrieval Options for Load Test Enabled Tests

For Load Test Enabled tests, the Select Link Source dialog box provides additional controls for the Input properties, when choosing the **Data source column** option.

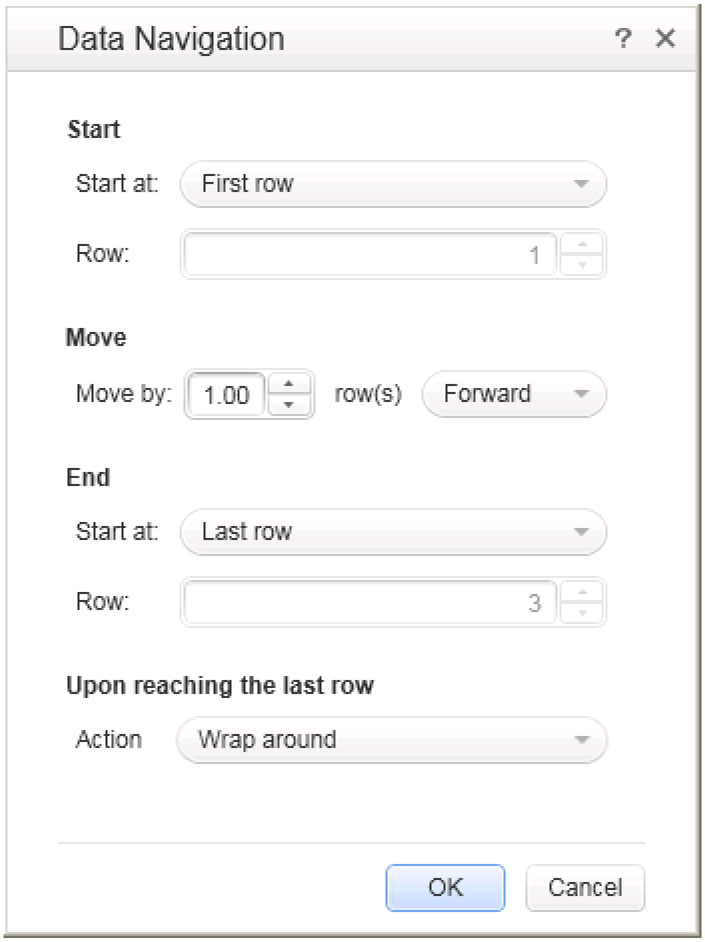
User interface elements are described below:

UI Elements	Description
Use a unique value for each Virtual User when load testing	Assigns a unique value to each Virtual User, from the data table, with the following properties: • **Data source parameter.** The parameter to use as a source of values for the selected input property. • **Update value on.** The frequency by which LoadRunner will update the values: **Each Iteration**, **Each Occurrence** (of the parameter), or **Once**. • **When out of values.**The action to do when reaching the end of the data table: ▪ **Abort Vuser.** End the test run for the Virtual User. ▪ **Continue Cyclic.** Return to the top of the data table and use the data in a cyclic manner. ▪ **Continue with Last.** Use the last value retrieved from the data table for all subsequent occurrences.
Allocate Virtual User values in the Controller	Allocates a specific amount of values from the data table for the Virtual User: • **Automatically allocate block size.** Automatically allocates a block size from the data based on the number of iterations and Virtual Users (only for Update value on **Each Iteration).** • **Allocate <amount> values for each Virtual User.** Allocates a specific number of values for each Virtual User. **Note:** Allocation is only possible when choosing Update value on **Each Occurrence** or **Each Iteration.**.

Data Navigation Dialog Box

Relevant for: API testing only

This dialog box enables you to set the data navigation properties for your test loop. You can set the direction in which to use the data, from where to begin, and the condition for selecting data.



Start

Start at: First row

Row: 1

Move

Move by: 1.00 row(s) Forward

End

Start at: Last row

Row: 3

Upon reaching the last row

Action Wrap around

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. In the Properties pane, link a property to a data source . 3. In the canvas, select the Test Flow or another test loop. 4. In the Properties pane, Open the **Data Sources** tab . 5. Select a data source in the list and click **Edit**.
Relevant tasks	"How to Set the Data Source Navigation Properties" on page 1825
See also	"Navigating Within a Data Source" on page 1813

User interface elements are described below:

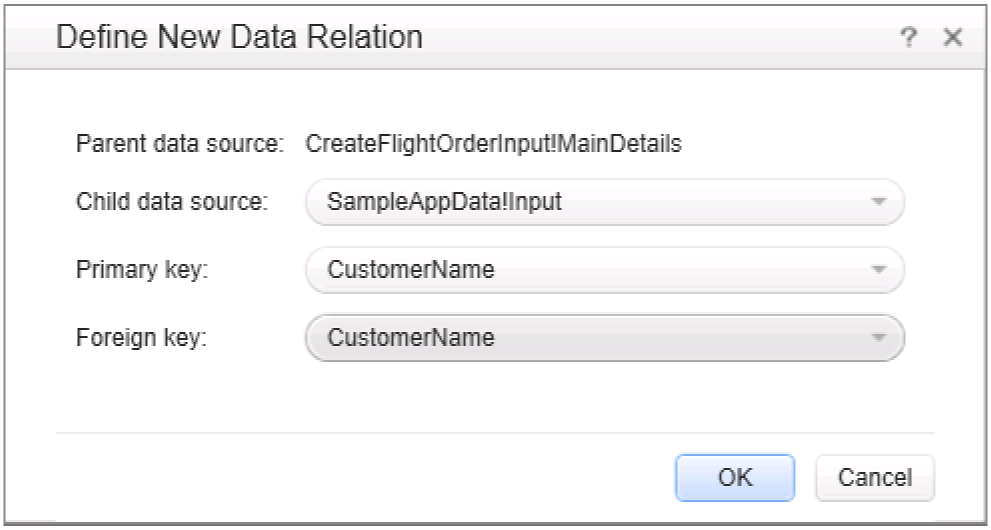
UI Elements	Description
Start	The location from where to begin taking data from the data source. • **Start at.** The row of data from which to begin: **First row**, **Last row**, **Specific row**, **Random row**, or **First row matching.** • **Row.** The row number from which to begin (only available when selecting **Specific row**). • **Condition.** The condition the data row must meet in order to start the looping (only available when selecting the **First row matching** option): ▪ **Column.** One of the data source's columns. ▪ **Comparison operator.** =, !=, >, >=, <, <=, Starts, Ends, Contains, or Regex (depending on the row's data type). ▪ **Value.** The value to be matched in order to use this row of data.
Move	The direction and amount of rows by which to advance within the data. • **Move by.** The number of rows to advance (not relevant for **Direction > To a random row**). • **Direction.** The direction in which to move: **Forward**, **Backward,** or **To a random row**.

UI Elements	Description
End	When to stop the looping through the data: **First row**, **Last row**, **Specific row**, **or First row matching**. • **Row.** The row number in which to end (only available when selecting **Specific row**). • **Condition.** The condition the data row must meet in order to end the looping (only available when selecting the **First row matching** option): ▪ **Column.** One of the data source's columns. ▪ **Comparison operator.** =, !=,>, >=, <, <=, Starts, Ends, Contains, or Regex (depending on row's data type). ▪ **Value.** The value to be matched in order to use this row of data. **Note:** This setting stops the looping only when the loop type is set to '**For Each**' (see "Loop Activity" on page 1652) and the current data source whose navigation settings you are editing, is the same data source that is linked to the loop. **Tip:** This setting is only relevant when the **Move** direction is set to **Forward** or **Backward**. For the **To a random row** option, each row is visited once in random order. The navigation ends after all of the rows were visited.
Upon reaching the last row - Action:(Move Forward)	The action to take when reaching the last row of data. • **Keep using that row.** Keep using the last row of data for all subsequent loops. • **Wrap around.** Start again from the first row of data.
Upon reaching the first row - Action:(Move Backward)	The action to take when reaching the first row of data. • **Keep using that row.** Keep using the first row of data for all subsequent loops. • **Wrap around.** Start again from the last row of data.

Define New/Edit Data Relation Dialog Box

Relevant for: API testing only

This dialog box enables you to define or edit a new data relation for a data source. The following is an example of the Define New Data Relation dialog box:



Define New Data Relation

Parent data source: CreateFlightOrderInput!MainDetails

Child data source: SampleAppData!Input

Primary key: CustomerName

Foreign key: CustomerName

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. Make sure you have at least one data source in the Data pane. 3. In the Data pane, select a data source. 4. In the Properties pane, open the **Data Source Properties** tab ▦. 5. Click **Add** or **Edit**.
Relevant tasks	"Create a new child relation" on page 1826

User interface elements are described below:

UI Elements	Description
Parent data source	The name of the parent data source, selected in the Properties pane (read-only).

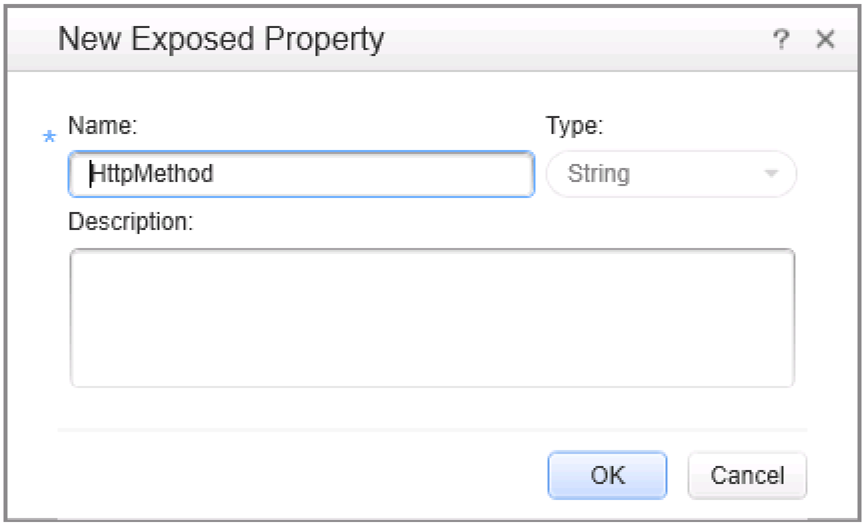
UI Elements	Description
Child data source	The sheet or table to use for the child data source.
Primary key	The column in the parent data source.
Foreign key	The column in the child data source to use in place of the primary key column.

New Exposed Property Dialog Box

Relevant for: API testing only

Note: This dialog box is relevant only if you are working withAPI tests created in UFT 11.51 or earlier or Service Test 11.51 or earlier.

This dialog box enables you to expose an internal HTTP property and make it available to other steps, from the wrapper. This is primarily useful for REST service steps with inner HTTP Requests.



* Name: HttpMethod

Type: String

Description:

)

To access	Do one of the following: **From within the Add/Edit REST Service dialog box:** • Define a REST service method. • In the right pane, click the **Custom Input/Checkpoints** tab. • Click within a property row. • Select **Expose as Input Property** or **Expose as Output Property** from the right-click menu. **From within the canvas:** • Add a REST method to the canvas. Expand it and select its inner **HTTP** step. • Open the Properties pane and click within a property row. • Select **Expose as Input Property** or **Expose as Output Property** from the right-click menu.
Relevant tasks	"How to a Create a REST Service" on page 1746
See also	"Add/Edit REST Service Dialog Box" on page 1772

The following elements are included:

UI Elements	Description
Name	A name for the exposed property.
Type	A drop down list of simple data types, such as String, Integer, and so forth. This field is read-only as it is determined by the exposed property's type.
Description	A description of the property.

Troubleshooting and Limitations - Using Data in API Tests and Components

Relevant for: API testing only

This section describes troubleshooting and limitations for working with data.

General

- If a specified data source is or becomes inaccessible, the test will not fail. The Errors pane and report, however, indicate that there was an error in retrieving the data.

- Keywords such as **#SKIP#**, and so forth, are not supported in the Input property grid.

  Workaround: Link to a data source that contains the keyword.

- When adding a referenced Excel data source, if the file requires special credentials (for example, a location on another domain or a drive requiring authentication), you must verify that the operating system will allow access to the file.

- Data sources with parent-child relationships, should not be accessed together in the same loop, unless the child data source is used for data-driving array elements. Accessing parent-child data sources in the same loop, may corrupt the test.

- Linking to file names is not supported for **HTTP Request** and **HTTP Receiver** activities that use the **File** type message body.

- For data sources with child relations: If you change the name the Key column in the child sheet, the Define New/Edit Data Relation dialog box does not reflect the new column name in the drop down lists.

Data Driving

- Data driving for JSON requests or responses, is only supported for Excel.

- Data driving for an Excel data source is only effective for the first 254 properties of a step. If a step has more than 254 properties, they will not be data-driven.

- Data driving for an Excel data source is only supported for property values consisting of 255 characters or less. If a property value has more than 255 characters, the data driving mechanism offers to truncate the string.

- When data driving a test step that has an array with two or more nested levels, the data driving engine only copies the first element of each array to the Excel data tables.

- Keyword data driving to an XML data source is not supported.

Chapter 59: Updating Services and Assemblies

Relevant for: API testing only

This chapter includes:

Concepts

Updating Web Services

Relevant for: API testing only

You can update Web services that exist in the Toolbox pane, from their original locations or from alternate locations.

When you update a service, UFT first compares the service and port names. If the port names match, UFT transfers all of the data from the updated service, including new security information. If the new version of the service contains a port that was not present in the original service, it adds it to the Toolbox pane, as an additional port for the service.

If the port paths (<service name:port name> combination) do not match, and security was set for on the port level, UFT opens the **Update Port Security Wizard**. For details, see the "Update Port Security Wizard" on page 1863.

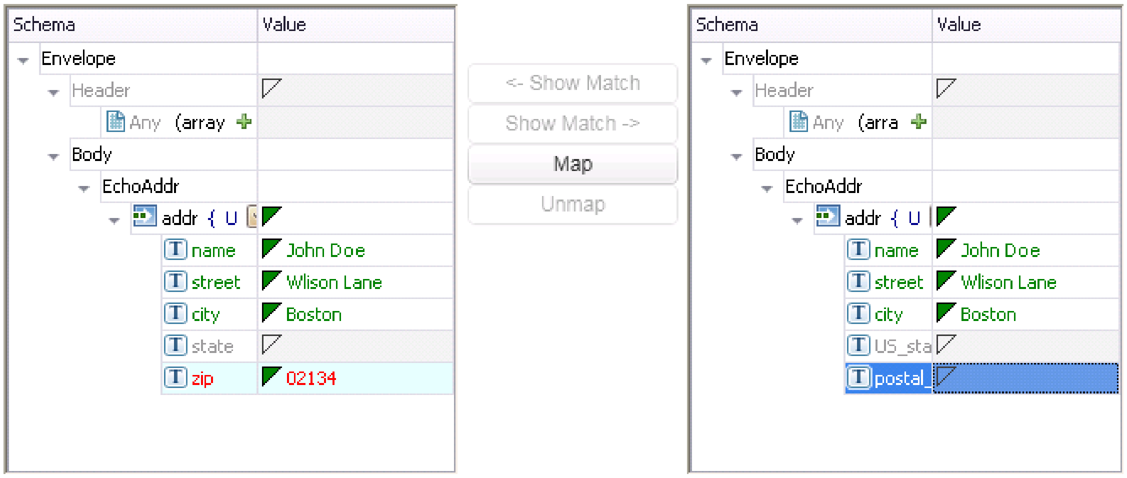
If HTTP transport information such as the endpoint was changed, UFT will update this automatically, as long as the port path is the same.

In the next step of the update, UFT compares the operations and properties of the current test steps. UFT tries to match up the properties that were assigned values between the original service and the updated version.

If the operation names do not match, the Update Step wizard enables you to map the new operation name to the original name.

If the operation names match, but the property names do not, the Update Step wizard enables you to map the new property names to the original ones. Matching properties are ones that have the same XPaths and types.

The wizard screen marks resolved properties in green and unresolved properties in red.



If you originally imported a service from ALM through Service Test Management, you must update the service in ALM—you will not be able to update the service through UFT.

When you update an RPC Web service, as long as the operation names match, the properties are marked as resolved. If the operation names cannot be resolved, UFT opens the wizard. If there is no operation available to map to the original, the wizard suggests that you delete the affected step.

For task details, see "How to Update a Web Service" on page 1857.

You can also update the data assigned to properties. When working with an Excel data source, you can update the data in the Data pane. For details, see "New/Change Excel Data Source Dialog Box" on page 255.

REST Service Conflicts

Relevant for: API testing only

Using REST services that you added to the Toolbox pane, you can create tests with minimal configurations. For details, see "How to a Create a REST Service" on page 1746.

If you modify a REST step's properties after incorporating it into your test, it will no longer match the method in the Toolbox. You may want to keep the change, or you may want to remove the conflicts so that your step will match the method defined in the toolbox.

You may encounter conflicts in the following areas:

- Adding or removing an input or output property
- Renaming an input or output property
- A change in the REST service URL
- A change in the HTTP request/response body type or schema
- Internal links between a REST property and its inner HTTP elements (if you are working with an API test created inUFT 11.51 or earlier or Service Test 11.51 or earlier
- HTTP methods

The Resolve REST Method Steps wizard enables you to view the conflicts and decide what to do with the conflicts in your test step—keep, remove, or map them. The wizard lets you resolve property-related conflicts. Other conflicts, such as discrepancies in the URL values, the HTTP body, and so forth, are resolved automatically.

For wizard details, see the "Resolve REST Method Steps Wizard" on page 1871.

For task details, see "How to Resolve Conflicts in a REST Service Test Step" on page 1860.

Updating SAP RFCs or IDocs

Relevant for: API testing only

You can update SAP RFCs and IDocs from their original location or from another connection or location.

The **Update** option automatically updates the item from its original location. UFT loads the information for the RFC/IDoc with the same names, from the original connection.

The **Update from** option lets you specify different connection information, or a name of a different RFC/IDoc on the same server.

If UFT detects an inconsistency between the original and updated RFC/IDoc names or their properties, it enables the Update Step wizard. This wizard lets you map the original and new items. For details, see "How to Update an SAP RFC or IDoc" on page 1861.

Tasks

How to Update a Web Service

Relevant for: API testing only

This task describes how to update a WSDL-Based Web Service that was already imported and incorporated into a test.

This task includes the following steps:

- "Prerequisites" below
- "Update the service " below
- "Run the Update Port Security wizard - optional" below
- "Run the Update Step wizard" on the next page

1. ### Prerequisites

   If you want to update the WSDL from ALM, make sure you have an open connection to the ALM server. For details, see the "ALM Connection Dialog Box" on page 737.

2. ### Update the service

   - To update a service from its original location, select its main node in the Toolbox pane. Choose **Update WSDL** from the context menu.
   - To update the service from a different location, or if the **Update WSDL** option is not available:

     i. Select the service's main node in the Toolbox pane.

     ii. Choose **Update WSDL from > URL or UDDI** or **Update WSDL from > File or ALM Application Component** from the right-click menu.

     iii. Navigate to the WSDL and click **OK**.

     iv. Accept any warning or informational pop-ups.

   > **Note:** The **Update WSDL** option may not be available if the user credentials changed or if the service was imported using Service Test or an earlier version of UFT.

3. ### Run the Update Port Security wizard - optional

If UFT detects a change in the port path (<service name:port name> combination) the **Update Port Security** Wizard opens. The wizard only opens if you configured security settings for the original service. Follow the steps of the wizard to resolve all of the Service/Port conflicts. For details, see "Update Port Security Wizard" on page 1863.

> **Note:**
>
> - The wizard imports all of the services defined in the WSDL, even those deleted manually before the update. To remove them from the Toolbox pane, delete them again manually, after the update.
> - If you manually remove a service with conflicts from the Toolbox pane, you will no longer be able to resolve the conflicts using the wizard.

4. **Run the Update Step wizard**

If a test step became invalid because an operation name changed or properties with values were changed, then the canvas marks the step with a warning marker.

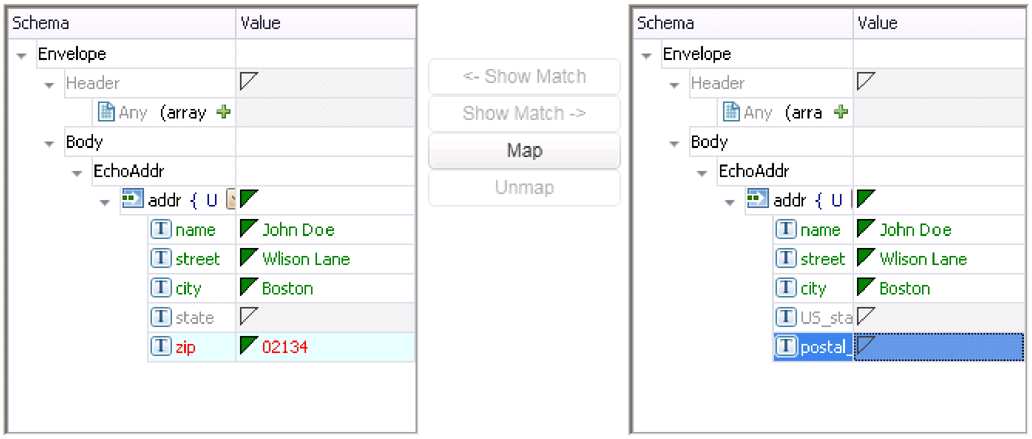
For details about the wizard, see the "Update Step/Activity Wizard" on page 1866.

a. Click the warning marker to display the message **The step must be resolved. Resolve step.** Click on the message text. The **Update Step Wizard** window opens.

Note that the step's properties become read-only, until you resolve them with the help of the wizard.

b. In the **Select Operation** page, click in the **New Operation** pane and select the operation that corresponds to the one in the **Original Operation** pane. Click **Next**. Properties for which conflicts were detected are highlighted in red.

c. In the **Update Input Properties** page, in the **Original Properties** pane, select a property for which there is a conflict, highlighted in red. In the right pane, **New Properties**, select a property to map.



Click **Map**. The wizard adds the mapping to the list in the bottom pane. Repeat this for all properties that you want to map. Click **Next**.

d. In the **Update Output Properties** page, select a property for which there is a conflict, highlighted in red. In the **New Properties** pane select the property to which you want to map. Click **Map**. The wizard adds the mapping to the list of mappings in the bottom pane. Repeat this for all properties that you want to map. Click **Next**.

e. Click **Finish** to save your changes and close the wizard.

How to Update a .NET Assembly

Relevant for: API testing only

This task describes how to update a .NET assembly with a newer version. The updating of .NET assembly is similar to the importing of an assembly described in "How to Import and Create a .NET Assembly API Test Step" on page 1760.

This task includes the following steps:

- "Select the assembly to update" below
- "Open the Import .NET Assembly dialog box" below
- "Import a .NET assembly" below
- "Handle warnings - optional" below
- "Modify custom code - optional" on the next page

1. Select the assembly to update

In the Toolbox pane, expand the **.NET Assemblies** node and select the assembly you want to update.

2. Open the Import .NET Assembly dialog box

Click the **Import .NET Assembly** button on the toolbar. For user interface details, see "Import .NET Assembly Dialog Box" on page 1774.

3. Import a .NET assembly

Click the **.NET Assembly Browser** tab. Click **Browse** and locate the .dll or .exe file. Click **Open** in the Open dialog box to add it to the **Selected References** list. Click **OK** to begin the update.

4. Handle warnings - optional

If the updated .NET assembly is missing types that are in use by existing activities, you will need to modify the step properties. The canvas displays warning icons adjacent to the steps whose properties use the missing type.

5. **Modify custom code - optional**

   If you wrote custom code in an event handler, you may need to modify the step properties. If the updated .NET assembly is missing types that are used by the custom code, make sure to modify the code to use only the types that are available.

How to Resolve Conflicts in a REST Service Test Step

Relevant for: API testing only

This task describes how to update a REST method step that was changed.

This task includes the following steps:

- "Prerequisites" below
- "Modify the step as required" below
- "Open the wizard" below
- "Run the wizard" below

1. **Prerequisites**

   Create one or more prototype methods for REST services. Drag them onto the canvas to create REST method steps. For details, see "How to a Create a REST Service" on page 1746.

2. **Modify the step as required**

   Modify the REST service step as required: add or remove properties, change property values, and so forth. If there are conflicts, the canvas will display warning icons next to the relevant steps.

3. **Open the wizard**

   - To resolve conflicts for the selected step, click the warning icon in the top right corner of the REST method step in the canvas. Select **This step should be resolved. Resolve step.** The Resolve REST Method Steps wizard opens.

   - To resolve conflicts for all steps created with the prototype, right-click the method in the Toolbox pane and select **Apply Changes to all Steps**.

4. **Run the wizard**

   a. Select the steps that you want to resolve (only relevant when opening the wizard through the Toolbox pane).

      The left pane, **Before Changes**, shows the step's properties before they were resolved. The right pane, **After Changes**, shows the step's properties after they were resolved.

b. Proceed with the wizard, resolving or keeping the detected differences.

For details about the wizard, see the "Resolve REST Method Steps Wizard" on page 1871.

How to Update an SAP RFC or IDoc

Relevant for: API testing only

This task describes how to update an SAP RFC or IDoc from its original or from a different location.

This task includes the following steps:

- "Update the RFC/IDoc from its original location" below
- "Update the RFC/IDoc from a different location" below
- "Run the Update Step wizard" below

Update the RFC/IDoc from its original location

To update an item from its original location, drill down to the actual RFC or IDoc in the Toolbox pane. This assumes that the location and name of the RFC or IDoc did not change. Choose **Update** from the context menu.

Update the RFC/IDoc from a different location

To update an RFC or IDoc from a different server, or from the same server but a different RFC or IDoc:

1. Select the RFC or IDoc in the Toolbox pane and choose **Update from** from the context menu. The Update <Item_Name> dialog box opens.

2. To change the connection or server information, select **Override connection** and specify the details. Alternatively, you can change the default connection information in the "SAP Connections Pane (Options Dialog Box > API Testing Tab)" described on page 568.

3. To select a different RFC or IDoc, search for it and select it in the bottom pane. For details, see the "Import from SAP/Update Dialog Box" on page 1729.

4. Click **Update**. Accept any warning or informational pop-ups.

Run the Update Step wizard

If a test step contains conflicts because an RFC or IDoc name or property changed, then the canvas marks the step with a warning marker.

1. Click the warning marker to display the message **The step must be resolved. Resolve step.** Click on the message text. The **Update Step Wizard** window opens.

   Note that the step's properties become read-only, until you resolve them with the help of the wizard.

2. In the wizard's **Select Activity** page, click in the **New item** area and select the operation that corresponds to the one in the **Original item** pane. Click **Next**. Properties for which conflicts were detected are highlighted in red.

3. If there are conflicted input properties, the **Update Input Properties** page opens. In the **Original Properties** pane, select a property for which there is a conflict, highlighted in red. In the right pane, **New Properties**, select a property to map. Click **Map**. The wizard adds the mapping to the list in the bottom pane. Repeat this for all properties that you want to map. Click **Next**.

4. If there are conflicted output properties, the **Update Output Properties** page opens. Select a property for which there is a conflict, highlighted in red. In the **New Properties** pane select the property to which you want to map. Click **Map**. The wizard adds the mapping to the list of mappings in the bottom pane. Repeat this for all properties that you want to map. Click **Next.**

5. Click **Finish** to save your changes and close the wizard.

   For details about the wizard, see the "Update Step/Activity Wizard" on page 1866.

Reference

Update Port Security Wizard

Relevant for: API testing only

This wizard enables you to update ports that became invalid as a result of an update action. The steps became invalid because the port path (<service name:port name> combination) changed. The wizard enables you to map old port names to new ones.

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the Solution Explorer, select an API test or component. 2. Import a WSDL using the **Import WSDL** toolbar button. 3. In the Toolbox pane, click on one of the service's ports and select **Security Settings** from the right-click menu. 4. In the Security Settings for Port dialog box, configure the port's security. 5. Select the service's parent node in the Toolbox pane and select **Update** (or **Update from)** from the right-click menu. **Note:** This wizard only opens if there is a discrepancy between the service or port names, and if the port's security settings were configured.
Relevant tasks	"How to Update a Web Service" on page 1857
Wizard map	This wizard contains: "Map Services and Ports Page" > "Finish Page"
See also	"Updating Web Services" on page 1854

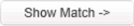
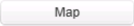
Map Services and Ports Page

Relevant for: API testing only

This wizard page enables you to map a port or service from the new WSDL, with the original WSDL's service name and port. This page is repeated for each of the services for which there is a conflict.

Important information	General information about this wizard is available here: "Update Step/Activity Wizard" on page 1866.
Wizard map	This wizard contains: **Map Services and Ports Page** > "Finish Page"

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

UI Elements	Description
	Shows the service/port in the **Available Services and Ports** pane, that is mapped to the service/port selected in the **Original Service and Ports** pane.
	Shows the service/port in the **Original Service and Ports** pane, that is mapped to the service/port selected in the **Available Services and Ports** pane.
	Maps the selected service/port in the **Original Service and Ports** pane, to a service/port in the **Available Services and Ports** pane. After you map a service/port, the grid displays it in green. **Note:** This button is enabled only if you select a red or black service/port in the **Original Service and Ports** pane, and a service/port in the **Available Services and Ports** pane.
	Removes the mapping of the selected service/port in the **Original Service and Ports** pane, from a service/port in the **Available Services and Ports** pane. **Note:** This button is only enabled if you select the mapped service/port in both panes. To find the mapped service/port, use the **Show Match** buttons.
<mapped property list>	A list of all the mapped services and ports—the ones mapped automatically by UFT and the ones mapped manually. **Tip:** Click on a set of services and ports to highlight them in the upper panes.
Available Services and Ports	A tree hierarchy of all of the available services and ports. The services and ports displayed in green text did not change in the update.

UI Elements	Description
Original Service and Ports	A tree hierarchy of the first service or port for which a conflict was found. As you click **Next**, the wizard proceeds to the next conflict. The wizard displays the services and ports in the following colors: **Green.** A service or port that did not change in the update or that was resolved through mapping. **Red.** A services or port that changed during the update and requires mapping. **Black.** A service or port that was resolved automatically during the update, but you chose to unmap it.

Finish Page

Relevant for: API testing only

This wizard page indicates whether you successfully resolved the conflicts between your original services and ports and those from the updated service.

Important information	General information about this wizard is available here: "Update Port Security Wizard" on page 1863.
Wizard map	This wizard contains: "Map Services and Ports Page" > Finish Page

User interface elements are described below:

UI Elements	Description
Ignores unresolved services/ports	Ignores unresolved conflicts. If you intend to remove the unresolved services, you can enable this option. **Tip:** To return to the wizard screens to resolve all of the services or ports, click **Back**.
Finish	Closes the wizard and applies your changes to the test. **Note:** Enabled when you successfully map all unresolved services and ports, or when you choose to ignore unresolved conflicts.

Update Step/Activity Wizard

Relevant for: API testing only

This wizard enables you to update test steps that became invalid as a result of an **Update from** action. The steps became invalid because the step/activity was removed or changed, or property names were modified. The wizard enables you to map old operations/activities and properties to new ones.

To access	1. Do one of the following: - Ensure that an API test or component is in focus in the document pane. - In the Solution Explorer, select an API test or component. 2. Use the **Tools** menu to import a Web service or an SAP RFC or IDoc. 3. Drag an operation/activity from the Toolbox pane onto the canvas and set input values. 4. Select the parent node and select **Update** or **Update from** in the shortcut menu. Accept any warning popup messages. 5. Click on the alert adjacent to the step ⊗. 6. Click on the text **The step must be resolved. Resolve step**. **Note:** The wizard will open only when there is a change in property names that were assigned values.
Relevant tasks	"How to Update a Web Service" on page 1857
Wizard map	This wizard contains: "Select Operation/Activity Page" > "Update Input Properties Page" > "Update Output Properties Page" > "Finish Page"
See also	"Updating Web Services" on page 1854

Select Operation/Activity Page

Relevant for: API testing only

This wizard page enables you to map an operation/activity from the new service/RFC or IDoc, with the original one. The **Next** button proceeds to the next conflict.

Important information	- General information about this wizard is available here: "Update Step/Activity Wizard" on the previous page. - The wizard may indicate that it could not find any compatible operations in the updated contract. This may be a result of different SOAP versions.
Wizard map	This wizard contains: **Select Operation/Activity Page** > "Update Input Properties Page" > "Update Output Properties Page" > "Finish Page"

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

UI Elements	Description
<search box>	**For Web services:** Enables you to filter the operations list. For example to display only those operations containing the term Get, type Get into the text field.
New operation/item pane	**For Web services:** A tree hierarchy of the new service, its ports, and its operations. **For SAP IDocs and RFCs:** An expansion of the new item in the toolbox showing the SAP Connection name, item type (RFC or IDoc), and item name.
Original operation/item	**For Web services:** A tree hierarchy of the operation in the selected test step. **For SAP RFCs and IDocs:** The original RFC or IDoc.
Apply to all steps of this type	Converts all test steps using the original operation/item, to steps using the new operation/item name. If you do not select this option, you will need to resolve the same conflict in other test steps, one-by-one. **Note:** This option is only enabled when you have multiple steps in the canvas that use the original operation/activity.

Update Input Properties Page

Relevant for: API testing only

This wizard page enables you to map new input properties with the original input properties.

Important information	General information about this wizard is available here: "Update Step/Activity Wizard" on the previous page.
Wizard map	This wizard contains: "Select Operation/Activity Page" > **Update Input Properties Page** > "Update Output Properties Page" > "Finish Page"

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

UI Elements	Description
Show Match ->	Shows the property in the **New Properties** pane, that is mapped to the property selected in the **Original Properties** pane.
<- Show Match	Shows the property in the **Original Properties** pane, that is mapped to the property selected in the **New Properties** pane.
Map	Maps the selected property in the **Original Properties** pane, to a property in the **New Properties** pane. After you map a property, the grid displays it in green. **Tip:** This button is enabled only if you select a red or black property in the **Original Properties** pane, and a property in the **New Properties** pane. **Note:** When mapping to an operation from another WSDL, the new WSDL must have the same SOAP version as the original.
Unmap	Removes the mapping of the selected property in the **Original Properties** pane, from a property in the **New Properties** pane. **Note:** This button is enabled only if you select the mapped properties in both panes. To find the mapped properties, use the **Show Match** buttons.
Original Properties	A tree hierarchy of all of the original step's input properties. The text color indicates the status: **Green.** Properties that did not change in the update or that were already mapped. **Red.** Properties that changed during the update and require mapping. **Black.** Properties that were resolved automatically during the update, but you chose to unmap them.
New Properties	A tree hierarchy of all of the new operation's input properties. The properties displayed in green text, are properties that did not change in the update.
Show only unmapped properties	Hides all non-red properties. The red properties are the ones that UFT was unable to resolve automatically during the update, and that you did not previously resolve.
<mapped property list>	A list of all the mapped properties—the ones mapped automatically by UFT and the ones mapped manually. **Tip:** Click on a set of properties to highlight them in the upper panes.

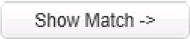
Update Output Properties Page

Relevant for: API testing only

This wizard page enables you to map new output properties with the original output properties. If the step has no output properties, the wizard skips this step.

Important information	General information about this wizard is available here: "Update Step/Activity Wizard" on page 1866.
Wizard map	This wizard contains: "Select Operation/Activity Page" > "Update Input Properties Page" > **Update Output Properties Page** > "Finish Page"

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

UI Elements	Description
 Show Match ->	Shows the property in the **New Properties** pane, that is mapped to the property selected in the **Original Properties** pane.
<- Show Match	Shows the property in the **Original Properties** pane, that is mapped to the property selected in the **New Properties** pane.
<mapped property list>	A list of all the mapped properties—the ones mapped automatically by UFT and the ones mapped manually. **Tip:** Click on a set of properties to highlight them in the upper panes.
Map	Maps the selected property in the **Original Properties** pane, to a property in the **New Properties** pane. After you map a property, the grid displays it in green. **Note:** This button is enabled only if you select a red or black property in the **Original Properties** pane, and a property in the **New Properties** pane.
New Properties	A tree hierarchy of all of the new operation's output properties. The properties displayed in green text, are properties that did not change in the update.

UI Elements	Description
Original Properties	A tree hierarchy of all of the original step's output properties. The text color indicates the status: • **Green.** Properties that did not change in the update or that were already mapped. • **Red.** Properties that changed during the update and require mapping. • **Black.** Properties that were resolved automatically during the update, but you chose to unmap them.
Show only unmapped properties	Hides all non-red properties. The red properties are the ones that UFT was unable to resolve automatically during the update.
Unmap	Removes the mapping of the selected property in the **Original Properties** pane, from a property in the **New Properties** pane. **Note:** This button is enabled only if you select the mapped properties in both panes. To find the mapped properties, use the **Show Match** buttons.

Finish Page

Relevant for: API testing only

This wizard page indicates whether you successfully resolved the conflicts between your original properties and the new ones.

Important information	General information about this wizard is available here: "Update Step/Activity Wizard" on page 1866.
Wizard map	This wizard contains: "Select Operation/Activity Page" > "Update Input Properties Page" > "Update Output Properties Page" > **Finish Page**

User interface elements are described below:

UI Elements	Description
Ignore unresolved properties	Ignores all unresolved input and output properties. If you intend to remove the step with the unresolved properties from your step, you can enable this option. **Tip:** To return to the wizard screens to resolve all of the properties, click **Back**.

UI Elements	Description
Finish	Closes the wizard and applies your changes to the test. **Note:** Enabled when you successfully map all unresolved properties, or when you choose to ignore unresolved properties.

Resolve REST Method Steps Wizard

Relevant for: API testing only

This wizard enables you to resolve differences between REST service steps and the prototype method upon which they were based. The wizard detects changes such as added, removed, or renamed properties and helps you resolve them. For details about the conflict types, see "REST Service Conflicts" on page 1855.

To access	Do one of the following: • Ensure that an API test or component is in focus in the document pane. • In the Solution Explorer, select an API test or component. As a prerequisite: 1. Create a prototype for a REST service method. For details, see "How to a Create a REST Service" on page 1746. 2. Drag one or more REST service methods from the Toolbox pane to the canvas. 3. Select a REST method in the Toolbox pane and choose **Edit Service** from the right-click menu. In the Edit REST Service dialog box, modify the method's properties and/or values. To start the wizard: • **For the selected step only:** Click on the alert in the bottom right corner of the REST method step ⚠ and click the text **The step should be resolved. Resolve step**. • **For all steps using the prototype method:** In the Toolbox pane, right-click the method and select **Apply Changes to all Steps**.
Relevant tasks	"How to Resolve Conflicts in a REST Service Test Step" on page 1860
Wizard map	This wizard contains: "Select Steps Page" > "Resolve Conflicts Page" > "Finish Page"

The wizard's global interface elements are described below:

UI Elements	Description
Cancel	Closes the wizard, discarding all resolutions that you made until this point.
Finish	Does the following: • Performs the automatic resolutions for remaining test steps whose conflicts were not resolved manually. • Ignores all remaining conflicting properties, by applying the **Keep** action to all of them. • Advances to the wizard's **Finish** page.
Next	Proceeds to the conflicts of the next step. The wizard displays a separate **Resolve Conflicts** page for each step.

Select Steps Page

Relevant for: API testing only

When you start the wizard from the Toolbox pane, it enables you to resolve conflicts for all steps that use the selected method. This wizard page enables you to include or exclude specific steps in which to apply the resolutions.

Important information	• General information about this wizard is available here: "Resolve REST Method Steps Wizard" on the previous page. • The ability to select the steps to resolve, is only available when you open the wizard from the Toolbox pane—not from the alert on the canvas. • The wizard only checks those REST steps that are based on the prototype selected in the Toolbox pane.
Wizard map	This wizard contains: **Select Steps Page >** "Resolve Conflicts Page" > "Finish Page"

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

UI Elements	Description
<test tree>	A tree hierarchy of the test steps. The following steps of the wizard will only affect the selected steps.

UI Elements	Description
Check All	Selects all test steps in which a conflict was detected.
Uncheck All	Clears the check box for all test steps in which a conflict was detected, excluding them from the wizard.

Resolve Conflicts Page

Relevant for: API testing only

This wizard page enables you to map new input/output properties with the original input/output properties.

Important information	- General information about this wizard is available here: "Resolve REST Method Steps Wizard" on page 1871. - The wizard repeats this page for each of the methods selected in the previous step.
Wizard map	This wizard contains: "Select Steps Page" > **Resolve Conflicts Page** > "Finish Page"

User interface elements are described below (unlabeled elements are shown in angle brackets). The **Keep**, **Remove**, and **Map** buttons relate to both the Input and Output properties.

UI Elements	Description
<conflict resolutions>	A list of all the resolved conflicts—the type of conflict, and the resolution. The text color indicates the conflict status: - **Green.** Conflicts that were resolved automatically or resolved by a **Keep**, **Map**, or **Remove** action. - **Red.** Conflicts that were not yet resolved or removed.

UI Elements	Description
Input Properties / Output Properties	A list of all of the step's input/output properties. • **Before changes.** A list of the step's original properties • **After changes.** A list of the step's properties after the conflicts were resolved. The text color indicates the property status: • **Green.** Properties that were resolved automatically or resolved by a **Keep**, **Map**, or **Remove** action. • **Red.** Properties for which a conflict was found, and not yet resolved. • **Black.** Properties for which no conflict was found.
Keep	Accepts the conflict for use within the test. The resulting test step's properties do not correspond to the prototype.
Map	Assigns the value of the selected property in the **Before changes** pane, to the property in the **After changes** pane After you map a property, the grid displays it in green. **Note:** This button is enabled only if you select a red or green property in the **Before changes** pane, and a black or green property in the **After changes** pane.
Remove	Removes the selected conflicting property from the current test step.

Finish Page

Relevant for: API testing only

This wizard page indicates whether you successfully resolved the conflicts between the prototype and the test steps.

Important information	General information about this wizard is available here: "Resolve REST Method Steps Wizard" on page 1871.
Wizard map	This wizard contains: "Select Steps Page" > "Resolve Conflicts Page" > **Finish Page**

User interface elements are described below:

UI Elements	Description
Finish	Closes the wizard and applies the resolutions to the test step. **Note:** If you click the **Finish** button before resolving the conflicts, the wizard applies a **Keep** action to all conflicts, and performs an automatic resolution for non-property related conflicts.

Troubleshooting and Limitations - Updating

Relevant for: API testing only

This section describes troubleshooting and limitations for updating services and assemblies.

REST Services

- When comparing a REST step to its prototype - a difference in the request/response body contents is not considered a conflict. Therefore, when resolving a REST step whose contents do not match the body contents of the prototype, the body contents will not be affected.

Chapter 60: Web Service Security

Relevant for: API testing only

This chapter includes:

Concepts

Setting Security Overview

Relevant for: API testing only

When building Web Service applications, there is a challenge in building scalable applications that are secure. You can secure Web Services by having the message sent over a secure transport, such as Secure Sockets Layer (SSL), or by applying security at the message level, also known as **WS-Security**.

For testing a secured service, answering the following questions will help you define your security scenario.

- Is there transport security, such as SSL? What is the HTTPS URL?
- Is basic authentication required?
- Is mutual authentication required?
- What type of security is required in the SOAP header?

Security Levels

UFT lets you set the security for a service on two levels—**port** or **operation**. If you define a security for a port, all of its methods use these settings, by default. When working in the canvas, you can override the default port security for a given test step and customize the security for a particular operation.

For task details, see "How to Set Security for a Specific Step" on page 1886.

Security Scenarios Overview

Relevant for: API testing only

UFT provides several built-in scenarios for configuring security in Web Service calls.

A security scenario represents a typical security implementation for a Web Service. It contains information such as authentication, encoding, proxy, certificates, and so forth.

A default **Web Service** scenario can be used for most Web services. It enables you to configure both transport and message-level security. UFT support for message-level security lets you manually configure the security elements such as tokens, message signatures, and encryption. For details, see "Web Service Scenario Overview" on the next page.

WCF scenarios enables you to configure security for HTTP or custom bindings and work with advanced specifications, such as **WS-SecureConversation**.

You use the **default Web Service scenario** for:

- Simple Web Services where no advanced standards are required.
- Web services using HTTP transport level security.
- Web services using message level security (WS-Security) for SOAP 1.1

You use a **WCF scenario** for:

- WCF Services that utilize advanced security and WS-Specifications.
- Web services using message level security (WS-Security) for SOAP 1.2

Such services can be written in various platforms such as WCF (Windows Communication Foundation), Metro (WSIT), and Axis2. UFT also supports proprietary standards and transports.

For details on the built-in security scenarios see one of following sections:

- "Web Service Scenario Overview" below
- "WCF Service (CustomBinding) Scenario Overview" on page 1881
- "WCF Service (Federation) Scenario Overview " on page 1882
- "WCF Services (WSHttpBinding) Scenario Overview " on page 1883

Web Service Scenario Overview

Relevant for: API testing only

The default Web Service scenario is based on the WS-Security specification. This scenario lets you place security credentials in the actual SOAP message.

When a SOAP message sender sends a request, the security credentials, known as **tokens**, are placed in the SOAP message. When the Web server receives the SOAP request, it does not need to send additional requests to verify the integrity of the sender. The server verifies that the credentials are authentic before letting the Web Service execute the application. By not having to go back to the source of the credentials, the application's scalability improves significantly.

To further secure Web Services, it is common to use digital signatures or encryption for the SOAP messages. Digitally signing a SOAP message verifies that the message has not been altered during transmission. Encrypting a SOAP message helps secure a Web Service by making it difficult for anyone other than the intended recipient to read the contents of the message.

The Security Settings dialog box provides the following tabs for the Web Service scenario: **HTTP**, **WS-Security**, and **WS-Addressing**. The **HTTP** tab lets you configure the transport security, the **WS-Security** tab handles the message-level security, and the **WS-Addressing** tab sets the addressing version.

You can use the following types of security:

Transport Level Security

The transport level security includes the authentication and proxy server information. You can also specify Keep Alive preferences and connection timeout. For user interface details, see "Result Tab (Properties Pane - API Testing)" on page 427.

Message Level Security

The **WS-Security** tab lets you set the message level security through tokens, signatures and encryption.

To support WS-Security, UFT enables you to create security tokens for your script. You can create multiple tokens and set their properties. After creating a token, you use it to sign or encrypt a SOAP message.

The Web Services security mechanism associates security tokens with messages. This mechanism supports several security token formats to accommodate a variety of authentication requirements. For example, a client might need to provide a proof of identity or a security certificate.

The available tokens are: **UserName**, **X509 Certificate**, **Kerberos**, **Kerberos2,** and **SAML**.

- **UserName.** The **User Name** token contains user identification information for the purpose of authentication: **User Name** and **Password**. You can also specify Password Options, indicating how to send the password to the server for authentication: **Text**, **None**, or **Hash**. and indicate whether to include a timestamp.

- **X509 Certificate.** This token is based on an X.509 certificate. To obtain a certificate, you can either purchase it from a certificate authority, such as VeriSign, Inc. or set up your own certificate service to issue a certificate. Most Windows servers support the public key infrastructure (PKI), which enables you to create certificates. You can then have it signed by a certificate authority or use an unsigned certificate.

- **Kerberos /Kerberos2.** (For Windows 2003 or XP SP1 and later) The Kerberos protocol is used to mutually authenticate users and services on an open and unsecured network. Using shared secret keys, it encrypts and signs user credentials. A third party, known as a KDC (Kerberos Key Distribution Center), authenticates the credentials. After authentication, the user may request a service ticket to access one or more services on the network. The ticket includes the encrypted, authenticated identity of the user. The tickets are obtained using the current user's credentials.

  The primary difference between the Kerberos and Kerberos2 tokens, is that Kerberos2 uses the Security Support Provider Interface (SSPI), so it does not require elevated privileges to impersonate the client's identity. In addition, the Kerberos2 security token can be used to secure SOAP messages sent to a Web Service running in a Web farm.

- **SAML Token.** SAML is an XML standard for exchanging security-related information, called assertions, between business partners over the Internet. The assertions can include attribute statements, authentication, decision statements, and authorization decision statements.

  SAML uses brokered authentication with a security token issued by STS (Security Token Service). The STS is trusted by the client and the Web Service to provide interoperable security tokens. SAML tokens are important for Web Service security because they provide cross-platform interoperability and a means of exchanging information between clients and services that do not reside within a single security domain.

Message Signatures and Encrypted Data

When you add a security token to a SOAP message, it is added to the SOAP message in the form of an XML element in the WS-Security SOAP header.

The message, however, is exposed and therefore requires additional security. This is especially true when the credentials, including the password, are sent in plain text as it is with role-based security.

The two methods used to secure the data are message signatures and message encryption:

- **Message Signatures.** Message Signatures are used by the recipients to verify that messages were not altered since their signing. The signature is in the form of XML within the SOAP message. The recipient checks the signature to make sure it is valid.

- **Message Encryption.** Although the XML message signature offers a mechanism for verifying that the message has not been altered since it was signed, it does not encrypt the SOAP message—the message is still plain text in XML format. To secure the message in order that it should not be exposed, you encrypt it, making it difficult for an intruder to view and obtain a user's password.

For task details, see "How to Set Security for a Specific Step" on page 1886.

For user interface details, see "Security Settings for Port <Port_Name> Dialog Box" on page 1900.

WCF Service (CustomBinding) Scenario Overview

Relevant for: API testing only

The **WCF Service (CustomBinding)** scenario enables the highest degree of customization. Since it is based upon the WCF CustomBinding standard, it enables you to test most WCF services, along with services on other platforms such as Java-based services that use the WS - <spec_name> specifications.

Use the WCF Service (CustomBinding) scenario to configure a scenario that does not comply with any of the predefined security scenarios. You can customize the transport and encoding settings:

- **Transport.** HTTP, HTTPS, TCP, or NamedPipe

- **Encoding.** Text, MTOM, or WCF Binary

You can also provide additional security information:

- **Authentication mode.** The type of authentication, such as None, AnonymousForCertificate, and so forth. The options are available from the drop down list.

- **Bootstrap policy.** For the SecureConversation authentication mode, you can specify a bootstrap policy.

- **Net Security.** The network security for TCP and NamedPipe type transports. Typical values are **None**, **Windows stream security**, or **SSL stream security** available from the field's drop down list. For services with HTTP transport, you should set the value to **None**. To enable SSL for HTTP, select **HTTPS transport**.

For task details, see "How to Customize Security for WCF Type Web Services" on page 1889.

For user interface details, see "WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 1911.

> **Note:** For WSE3 security configurations, use the WCF Service(CustomBinding) Scenario. For details, see "How to Customize Security for WCF Type Web Services" on page 1889.

WCF Service (Federation) Scenario Overview

Relevant for: API testing only

In the **WCF Service (Federation)** scenario, the client authenticates against the STS (Security Token Service) to obtain a token. The client uses the token to authenticate against the application server.

Therefore, two bindings are needed, one against the STS and another against the application server.

You define the bindings in two stages:

- Provide security details for the application server's security scenario in the following areas:
  - **Server.** The transport and encoding methods.
  - **Security.** The authentication mode, such as IssuedToken, SecureConversation, and so forth.
  - **Identity.** Information about the server certificate and expected DNS.
  - **STS.** Settings related to the STS, such as the endpoint address and binding.
- Define an STS binding in the **Referenced binding** field.

For task details, see "How to Customize Security for WCF Type Web Services" on page 1889.

For user interface details, see "WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 1911.

WCF Services (WSHttpBinding) Scenario Overview

Relevant for: API testing only

> **Note:** The **WCFService (WSHttpBinding)** scenario only supports the testing of WCF services which utilize *wsHttpBinding* and incorporate some level of security. To test WCF services that use *wsHttpBinding* but have no security, use the **WCFService (CustomBinding)** scenario.

In the **WCFService (WSHttpBinding)** scenario, you can select from several types of authentication: None, Windows, Certificate, or Username (message protection).

No Authentication (Anonymous)

In this scenario, the client uses the server's certificate to encrypt a message; there is no client authentication. Use this scenario to test Web Services where the:

- Client uses the server's X.509 certificate for encryption.
- Client is not authenticated.
- Communication may utilize advanced standards such as secure conversation or MTOM.

Windows Authentication

This scenario uses Windows Authentication. If you are testing a WCF service that has not been customized and uses the default configuration, use this type of scenario.

Use this scenario to test Web Services where the:

- Client and server use Windows authentication.
- Security is based on Kerberos or SPNEGO negotiations.
- Communication may utilize advanced standards such as secure conversation or MTOM.

Certificate Authentication

In this **WCF WSHttpBinding** scenario, the client uses the server's X.509 certificate to encrypt the message and its own certificate for a signature.

Use this scenario to test Web Services where the:

- Client uses the server's X.509 certificate for encryption.
- Client uses its own X.509 certificate for signatures.
- Communication may utilize advanced standards such as secure conversation or MTOM.

Username Authentication (Message Protection)

In the **WCF WSHttpBinding** scenario, the client uses the server's X.509 certificate to encrypt the message, and sends a user name and password to authenticate itself.

Use this scenario to test Web Services where the:

- Client uses the server's X.509 certificate for encryption.
- Client is authenticated with a username and password.
- Communication may utilize advanced standards such as secure conversation or MTOM.

For all of the authentication types, you can apply advanced settings as described in "Advanced Security Settings" below.

For user interface details, see "WCF Service (WSHttpBinding) Scenario (Security Settings for Port <Port_Name>) Dialog Box)" on page 1913.

Advanced Security Settings

Relevant for: API testing only

This scenario's settings let you customize For a **WCFService** scenario, you can add additional advanced security settings, including **Encoding**, **Advanced Standards**, **Security**, or **HTTP and Proxy**. Not all settings are relevant for each of the scenarios.

For details, see the "Advanced Settings Dialog Box" on page 1916.

Tasks

How to Set Security for a Web Service on the Port Level

Relevant for: API testing only

This task describes how to configure security settings for a Web service on the port level. You can override this by modifying the settings for a specific test step. For details, see "How to Set Security for a Specific Step" on the next page.

This task includes the following steps:

- "Prerequisites" below
- "Open the Security Setting dialog box" below
- "Load existing settings - optional" below
- "Create a new security scenario" below
- "Save the settings - optional " on the next page

1. ### Prerequisites

   Import at least one Web service.

2. ### Open the Security Setting dialog box

   Right-click a Web service's port node in the Toolbox pane and select **Security Settings**.

3. ### Load existing settings - optional

   a. In the Security Settings dialog box, To load an existing set of UFT security settings, click **Import** the .stss (Service Test Security Scenario) file.

   b. Navigate to a .stss (Service Test Security Scenario) file and load it into your test. The settings in the Security Settings dialog should match the settings saved in the security settings file.

4. ### Create a new security scenario

   Use the Security Settings dialog box to create a new security scenario or modify a loaded one:

   a. In the Service Details dropdown, select a scenario type.

   b. Configure the security settings for the selected scenario. For details, see the "Security Settings for Port <Port_Name> Dialog Box" on page 1900.

   c. For Web Service type scenarios, add tokens, signatures, and encryption. For details, see "Web Service Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page

1902.

d. For WCF security scenarios, add the security information. For details on these settings, see "WCF Service (Custom Binding) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 1909, "WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 1911, or "WCF Service (WSHttpBinding) Scenario (Security Settings for Port <Port_Name>) Dialog Box)" on page 1913.

5. Save the settings - optional

Click **Save** to save the settings to an .stss file.

How to Set Security for a Specific Step

Relevant for: API testing only

This task describes how to set security properties for an individual test step.

This task includes the following steps:

- "Add a step to which security can be applied" below
- "Open the Security Setting tab" below
- "Enable the step's security details" below
- "Specify a scenario type" below
- "Configure the security settings" below

1. Add a step to which security can be applied

Drag a Web Service operation or SOAP Request activity onto the canvas.

2. Open the Security Setting tab

Open the **Security Settings** tab 🔒 in the Properties pane.

3. Enable the step's security details

Clear the **Use the port's security settings** option.

4. Specify a scenario type

Select a scenario from the **Service Details** dropdown list.

5. Configure the security settings

Configure the security settings for the selected scenario. For task details, see one of the following sections:

- "How to Set Security for a Standard Web Service" below
- "How to Customize Security for WCF Type Web Services" on page 1889

To see the request and response before and after WSE2 (non-WCF) security is applied, view the Output pane after the test run.

How to Set Security for a Standard Web Service

Relevant for: API testing only

This task describes how to configure security settings for a standard Web Service. This mode lets you define the HTTP transport information and security elements such as tokens.

For details about standard Web service security, see "Web Service Scenario Overview" on page 1879.

This task includes the following steps:

- "Open the Security Settings dialog box or tab" below
- "Create a Web Service scenario" below
- "Configure the HTTP settings" on the next page
- "Define security elements (optional)" on the next page
- "Configure the WS-Addressing (optional)" on the next page

1. Open the Security Settings dialog box or tab

- To set security on a port level, right-click a Web Service port in the Toolbox and choose **Security Settings**.
- To set security for a specific Web Service step already on the canvas, select the step and open the **Security Settings** tab in the Properties pane. Clear the **use the port's security settings** option.
- For a SOAP Request step, click the **Security Settings** tab in the Properties pane.

For details on the available settings, see the "Security Settings for Port <Port_Name> Dialog Box" on page 1900.

2. Create a Web Service scenario

In the Security Settings dialog box, select **Web Service** from the **Service Details** dropdown list (default).

3. Configure the HTTP settings

In the main window of the Security Settings dialog box, select the **HTTP** tab, and set the transport and proxy information. For details on the transport and proxy settings, see the "HTTP tab" on page 1903.

4. Define security elements (optional)

In the main window of the Security Settings dialog box, click the **WS-Security** tab. Add tokens, message signatures, and encryption.

- To add a token, click the **Security Tokens** button and select a token type. In the lower pane, provide the token details.

  For details on the different types of tokens, see "Message Level Security" on page 1880.

  > **Note:** When adding a SAML token, if you have the complete SAML token string, you can paste it directly into the **AssertionIDReference** field. If you do not have the complete token string and you want to configure the token manually, make sure that the first row in the schema contains the **Assertion** property—not the **AssertionIDReference**. You can change this using the grid's drop down menu.

- To add a message signature, click the **Add Message Signature** button and select a token in the **Signing token** dropdown list. In the lower pane, provide the other required information.

  > **Note:** You must add a token before adding a message signature.

- To add a message encryption (you must first add at least one token), click the **Add Message Encryption** button and select the token for the encryption from the **Encryption token** box. Provide any other required information or accept the defaults.

  > **Note:** You must add a token before adding message encryption.

- Organize the security elements in their order of priority. Use the **Up** and **Down** arrows to set the priorities. The basic order is tokens, followed by message signatures, and then encryption. In addition, your service may also require a specific order for the tokens.

- Indicate whether or not to include a timestamp when sending the token, signature, or encryption to the server by selecting or clearing the **Exclude Timestamp** option.

For details on the available options for tokens, message signatures, and message encryption, see "WS-Security tab" on page 1904.

5. Configure the WS-Addressing (optional)

a. In the main pane of the Security settings dialog box, click the **WS-Addressing** tab.

b. In the WS-Addressing tab, select the relevant version or None if WS-Addressing is not used.

c. In the **Reply to** field, provide an alternate destination.

For user interface details see the "Security Settings for Port <Port_Name> Dialog Box" on page 1900.

How to Customize Security for WCF Type Web Services

Relevant for: API testing only

This section describes how to customize the security settings for Web services using WCF.

Note: This task is part of a higher-level task. For details, see "How to Set Security for a Web Service on the Port Level" on page 1885 or "How to Set Security for a Specific Step" on page 1886.

This section describes:

- "Prerequisite - create a WCF scenario" below
- "Configure the settings for a Web Service using WSHTTPBinding" on the next page
- "Configure the settings for a Web Service using CustomBinding" on the next page
- "Configure the settings for a WCF Federation Web service" on the next page
- "Configure the settings for a WCF service using netTcp or namedPipe transport" on page 1891
- "Configure the settings for a Web service using WSE3 security configuration with a server certificate" on page 1891
- "Configure the settings for WCF service using mutual certificate authentication" on page 1892
- "Configure the settings for a WCF scenario using binding with TCP transport to require an X.509 client certificate" on page 1893

Prerequisite - create a WCF scenario

- For port level security, right-click a service's port in the Toolbox pane and select **Security Settings**.
- For step level security, open the **Security Settings** tab in the Properties pane. Clear the **Use the port's security settings** option.

Select the type of **WCF Service**from the **Service Details** dropdown list.

Configure the settings for a Web Service using WSHTTPBinding

WSHttpBinding is one of the most popular bindings in WCF. In order to use this binding, select the **WCFService (WSHttpBinding)** scenario from the **Service Details** dropdown list.

1. In the Client authentication type dropdown list, choose a client credential type to use in your binding—Windows, Certificate, or Username. This value corresponds to the **MessageClientCredentialType** property of the WCF's WSHttpBinding parameter.

   Windows authentication is the most common value for a WCF services. If you are using the WCF default settings for your service, use this option.

2. Define the security settings for your authentication type. The available options differ per authentication type.

   > **Note:** For some scenarios you should indicate whether to use the WCF proprietary negotiation mechanism to get the service credentials.

3. Click **Advanced** to control the usage of a secure session. For details on the available Advanced security properties, see "Advanced Settings Dialog Box" on page 1916.

Configure the settings for a Web Service using CustomBinding

1. In the Security Settings dialog box, select the **WCFService (Custom Binding)** scenario.

2. In the main pane of the Security Settings dialog box, set the Web service security options, including:

   - **Transport** type
   - **Encoding**
   - **Authentication mode** for the Web service
   - **Net security** type
   - The **identities** for the custom bindings and authentication certificate
   - The **client user** information for the "user" who would access the Web service

Configure the settings for a WCF Federation Web service

1. In the Security Settings dialog box, select the WCFService Federation scenario from the Service Details dropdown list.

2. Provide the service and security transport details, including:

   - **Transport** type
   - **Encoding**

- **Authentication mode** for the Web service
- **Bootstrap policy** for the Web service
- The **identities** for the custom bindings and authentication certificate
- **STS** (Security Token Service) settings

Note: You must to define the communication properties for both the STS and the application server

Configure the settings for a WCF service using netTcp or namedPipe transport

1. In the Security Settings dialog box,select the **WCFService (Custom Binding)** scenario from the **Service Details** dropdown list.
2. Configure the transport to **TCP** or **NamedPipe**.
3. Set the other security settings as described in "Configure the settings for a Web Service using CustomBinding" on the previous page.

Configure the settings for a Web service using WSE3 security configuration with a server certificate

1. Create a new test and import the WSDL containing the W3E3 service.
2. Drag a method from the Web service to the canvas.
3. Select the **Security** tab in the Properties pane or right-click the Web service node in the Toolbox and select Security Settings.
4. In the Security Settings dialog box, select the **WCFService (Custom Binding)** scenario.
5. In the main pane of the Security Settings dialog box, set the Transport to **HTTP**, and the Encoding to **Text**.
6. In the Identities section, enter a username and password.
7. Click the **Browse** button adjacent to the Server Certificate field and specify the **Store Location**, **Store Name** and **Search text** (optional). Click **Find**, select the certificate, and click **Select**.
8. Provide the **Expected DNS**.

9. Click the **Advanced** button and configure the following settings in the Advanced Settings dialog box:

   a. In the **Encoding** tab: Set the **WS-Addressing** version appropriately

   b. In the **Security** tab, set the following options:

      - **Enable secure session:** Enabled

      - **Negotiate service credentials:** Enabled

      - **Protection level:** Encrypt and Sign

      - **Message protection order:** Sign Before Encrypt

      - **Message security version:** WSSecurity11WSTrustFebruary2005WSSecureConversationFebruary2005 (first entry)

      - **Require Derived keys:** Enabled

      For all other fields, use the default settings.

Configure the settings for WCF service using mutual certificate authentication

The following procedure describes how to set up a security scenario for mutual certificates and how to comply with a WSE3 security configuration.

1. In the Security Settings dialog box, select the **WCFService (CustomBinding)** scenario from the **Scenario Details** dropdown list.

2. Set the **Transport** to HTTP, and the **Encoding** to Text.

3. Set the authentication mode to MutualCertificate.

4. In the **Identities** section, select the server and client certificates. For details, see "Select Certificate Dialog Box" on page 1923.

5. Provide the **Expected DNS**.

6. Click the Advanced button and configure the following settings in the Advanced Settings dialog box:

   a. **Encoding** tab—**WS-Addressing**: WSA 04/08 (for a WSE3 security configuration).

   b. **Security** tab—**Require Derived keys**: Disabled

   For all other fields, use the default settings.

Configure the settings for a WCF scenario using binding with TCP transport to require an X.509 client certificate

The following procedure describes how to configure a WCF custom scenario to require an X.509 client certificate in **nettcp.**

1. In the Security Settings dialog box, select the **WCFService (Custom Binding)** scenario from the **Service Details** dropdown list.

2. Set the **Transport** to TCP and the **Net Security** to SSL stream security.

3. In the Properties pane, open the Events tab ⚡ .

4. IIn the events list, select the BeforeApplyProtocolSettings event. Click in the **Handler** column and select **Create a default handler** from the drop-down.

5. In the TestUserCode.cs file, locate the **TODO** section of the code and add the following definitions.

   ```
   var wcf = (HP.ST.Ext.CommunicationChannels.Models.WcfChannelBinding)args[1];
   var ssl =
     (HP.ST.Ext.CommunicationChannels.Models.WcfSslStreamSecurityChannel)wcf.
       Protocols.Channels[1];
   ssl.RequireClientCertificate = true;
   ```

   For all other fields, use the default settings.

6. Save the test and run it.

How to Set Common Web Services Security Scenario Properties

Relevant for: API testing only

This section illustrates how to set several common security scenarios properties. The examples apply when using the default Web Service security scenario.

You can set security for all operations in a Web service port or for a specific step in your test.

> **Note:** This task is part of a higher-level task. For details, see "How to Set Security for a Web Service on the Port Level" on page 1885 or "How to Set Security for a Specific Step" on page 1886.

This task includes the following:

- "Authenticating with a Username Token" on the next page
- "Signing with an X.509 Certificate" on the next page

- "Encrypting with a Certificate" on the next page
- "Authenticating with a Username Token and Encrypting with an X.509 Certificate" on page 1896
- "Encrypting and Signing a Message" on page 1896

Authenticating with a Username Token

To send a message level username/password token (a UserName token):

1. In the Security Settings dialog box, select the **Web Service** scenario from the **Service Details** dropdown list.
2. In the main part of the Security Settings dialog box, click the **WS-Security** tab.
3. In the WS-Security tab, click the **Add Token** button and add a **Username** token.
4. In the lower pane, customize the token details, such as username and password.

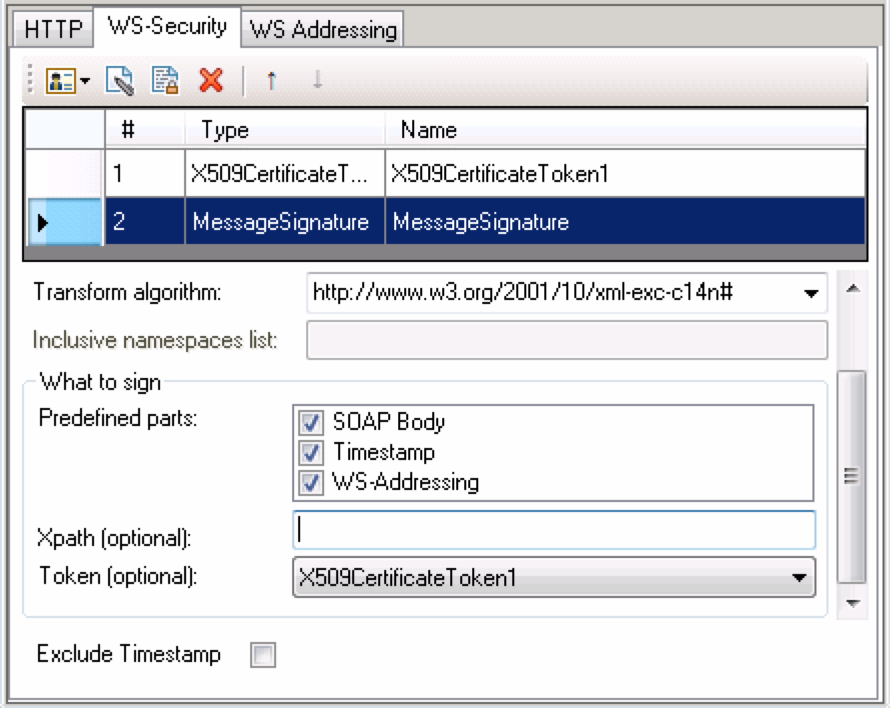
Signing with an X.509 Certificate

1. In the Security Settings dialog box, select the **Web Service** scenario from the **Service Details** dropdown list.
2. In the main part of the Security Settings dialog box, click the **WS-Security** tab.
3. In the WS-Security tab, click the **Add Token** button and select **X509 Certificate** from the dropdown list
4. In the lower pane, enter the token name.
5. Click the **Browse** button to navigate to your certificate file. The certificate must be installed in the Windows certificate store.
6. Select a **Reference type**. Since this token is used for a signature, the most common type is BinarySecurityToken.
7. Click the **Add Message Signature** button .
8. In the **Signing Token** dropdown list, select the token you entered in the previous steps.
9. To sign a specific element with the certificate, scroll down to the **XPath** field and provide an XPath expression.

   You cannot use an XPath expression to sign a timestamp or token that is under the security element of a SOAP request.

   - To sign a **SOAP Body**, **Timestamp,** or **WS-Addressing,** select the check box in the **Predefined parts** area.

- To sign tokens within the security element, select a token in the **Token (optional)** field, in the **What to sign** area.



	#	Type	Name
	1	X509CertificateT...	X509CertificateToken1
▶	2	MessageSignature	MessageSignature

Transform algorithm: http://www.w3.org/2001/10/xml-exc-c14n#

Inclusive namespaces list:

What to sign

Predefined parts: ☑ SOAP Body ☑ Timestamp ☑ WS-Addressing

Xpath (optional):

Token (optional): X509CertificateToken1

Exclude Timestamp □

Note: The certificate needs to be installed in the Windows certificate store. In the example above, you need to set the actual store name, store location, and subject name of your certificate.

Encrypting with a Certificate

To encrypt a message using the service's X.509 certificate:

1. In the Security Settings dialog box, select the **Web Service** scenario from the **Service Details** dropdown list,

2. In the main part of the Security Settings dialog box, select the **WS-Security** tab.

3. In the WS-Security tab, click the **Add Token** button and select **X509 Certificate** token from the **Security Token** drop down list.

4. Enter token name.

5. Since this token is used for encryption, use Reference as the **Reference type**.

6. Click the **Add Message Encryption** button . In the drop down list, select the token you created in the previous steps.

7. Scroll down to the **XPath** field. Enter an XPath expression to the elements to encrypt, for example: // *[local-name(.)='Body'].

Authenticating with a Username Token and Encrypting with an X.509 Certificate

The following section describes how to send a **Username** token to the service and encrypt it with the server's **X.509** certificate:

1. In the Security Settings dialog box, select the **Web Service** scenario from the **Service Details** dropdown list.

2. In the main part of the Security Settings dialog box, select the **WS-Security** stab.

3. In the WS-Security tab, click the **Add Token** button and select **Username Token** from the **Security Token** drop down list.

4. In the lower pane, Provide the token details for the Username token.

5. Click the **Add Token** button again and select **X509 Certificate Token** from the **Security Token** drop down list.

6. In the lower pane, enter the token details to reference the server's public certificate. Since this token is used for encryption, use Reference as the **Reference type**.

7. Click the **Add Message Encryption** button . In the drop down list, select the X.509 token you created in the previous steps.

8. To encrypt a specific message, scroll down to the **XPath** field. Enter an XPath expression, for example:, // *[local-name(.)='Body'].

Encrypting and Signing a Message

This example shows how to sign a message using a private key and then encrypt it using the service's public key.

1. In the Security Settings dialog box, select the **Web Service** scenario from the **Service Details** list.

2. In the main part of the Security Settings dialog box, select the **WS-Security** tab.

3. In the WS-Security tab, click the **Add Token** button and select **X509 Certificate Token** from the **Security Token** drop down list.

4. Enter the token details to reference your private certificate. Select BinarySecurityToken as a **Reference type**, since this token is used for a signature.

5. Click the **Add Token** button again and select **X509 Certificate Token** from the **Security Token** drop down list to add another X.509 token.

6. Enter the token details to reference your private certificate. Select Reference as a **Reference type**, since this token is used for a encryption.

7. Click the **Add Message Signature** button . In the drop down list, select the first X.509 token you created.

8. Click the **Add Message Encryption** button . In the drop down list, select the second token you created.

How to Test Web Services that use WS-Security or SSL

Relevant for: API testing only

This section provides step for setting up web services that use WS-Security or SSL protocols.

> **Note:** This task is part of a higher-level task. For details, see "How to Set Security for a Web Service on the Port Level" on page 1885 or "How to Set Security for a Specific Step" on page 1886.

This section includes:

- "How do I test a Web Service that uses SSL?" below
- "How do I test a Web Service that requires Windows authentication at the HTTP level?" below
- "How do I test a Web Service that uses WS-Security?" on the next page
- "How do I configure the low-level details of my WS-Security tokens?" on the next page

How do I test a Web Service that uses SSL?

Testing a secure site does not require any special configuration. If your service URL begins with https, SSL is automatically used.

If in addition to SSL you are also using message-level security (for example for a username) then you must configure the security for the message separately.

You can use the basic Web Services security scenario and specify the message-level security such as tokens and signatures. You can also use the **WCFService (Custom Binding)** scenario, or the **WCFService(WSHttpBinding)** scenario with the transport credentials.

How do I test a Web Service that requires Windows authentication at the HTTP level?

1. Select the **Web Service** scenario from the **Scenario Details** list.

2. Set the service security settings as described in "How to Set Security for a Standard Web Service" on page 1887.

How do I test a Web Service that uses WS-Security?

Use the **Web Services** security scenario and open the **WS-Security** tab. Add the message-level security such as tokens, signatures, and encryption. For details, see "How to Set Security for a Standard Web Service" on page 1887.

How do I configure the low-level details of my WS-Security tokens?

In most cases, you can configure the low-level details as described in "Advanced Settings Dialog Box" on page 1916.

How to Set up Advanced Standards Testing

Relevant for: API testing only

This section provides guidelines for using UFT in advanced standards testing.

This section includes:

- "How do I test a Web Service that uses MTOM?" below
- "How do I change the WS-Addressing version of a service?" below
- "How do I enable support for a service or activity that uses 256-bit SSL encoding?" on the next page

How do I test a Web Service that uses MTOM?

1. Select the **WCFService (Custom Binding)** scenario from the **Service Details** list.
2. Configure the **Encoding** to MTOM.

   If your service requires advanced settings, click the **Advanced** button. For details on the available options, see the "Advanced Settings Dialog Box" on page 1916.

For more details about the scenario, see "How to Customize Security for WCF Type Web Services" on page 1889.

How do I change the WS-Addressing version of a service?

1. Select the **Web Service** scenario from the **Service Details** list.

   > **Note:** If your service uses WCF, use the appropriate scenario and configure the addressing version from the Advanced Settings dialog box's **Encoding** tab. For details, see the "Advanced Settings Dialog Box" on page 1916.

2. Click the **WS-Addressing** tab and select a version.

How do I enable support for a service or activity that uses 256-bit SSL encoding?

Change the SSL cipher order in Windows Vista so that AES256 precedes AES128 in the cipher list.

> **Tip:** Check with an IT professional before performing the following actions.

To change the cipher order:

1. Type **gpedit.msc** at a command prompt to open your group policy editor.
2. Choose **Computer Configuration > Administrative Templates > Network > SSL Configuration Settings**.
3. Open the only item—**SSL Cipher Suite Order**.
4. Select **Enabled**.
5. The first item in the list is TLS_RSA_WITH_AES_128_CBC_SHA

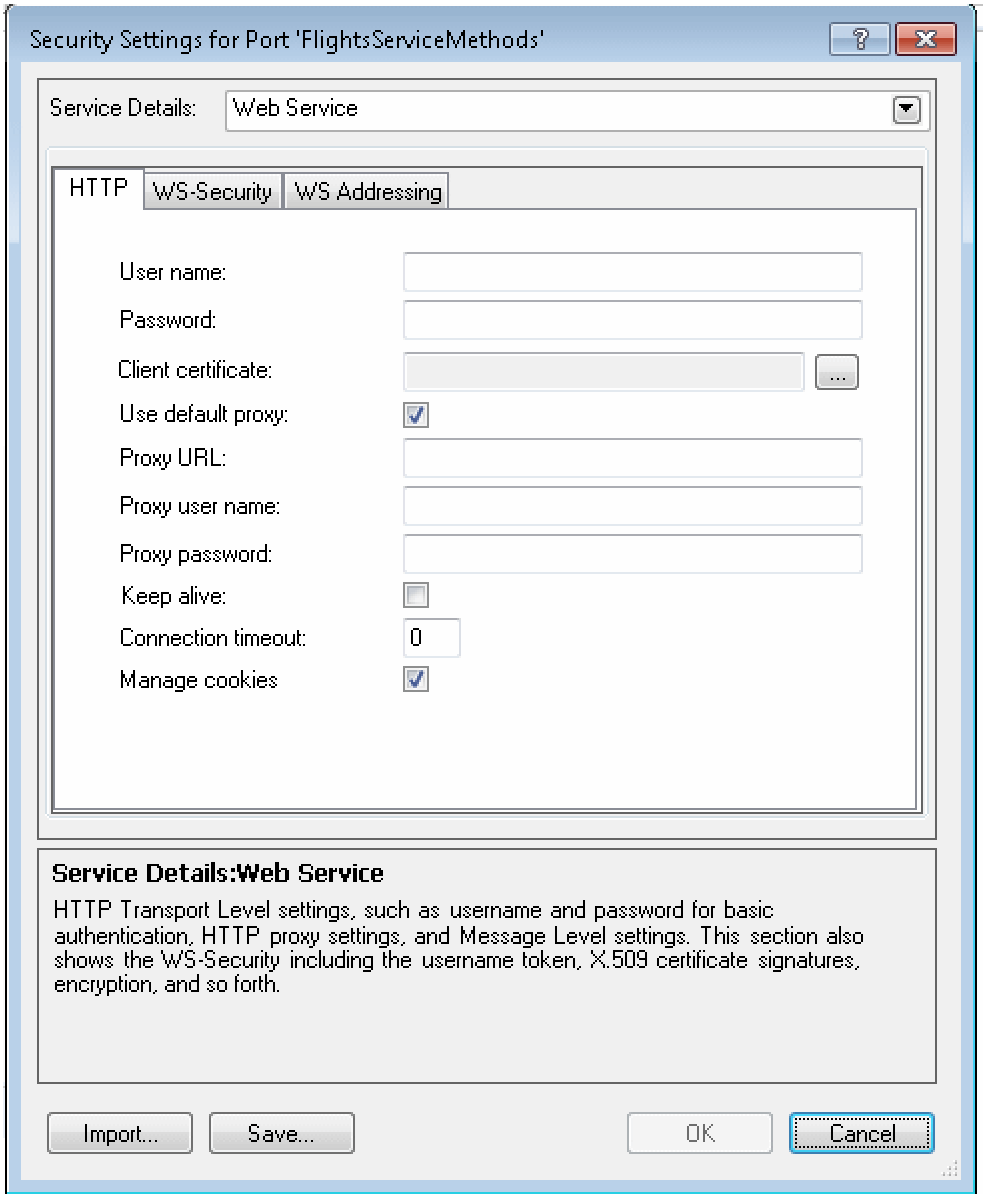
   The second item is TLS_RSA_WITH_AES_256_CBC_SHA
6. Change the first 128 to 256. Then move the cursor forward and change the 256 to 128.
7. Move the cursor through the list and change the cipher priorities as in the above step.
8. Close the group policy editor and reboot.

Reference

Security Settings for Port <Port_Name> Dialog Box

Relevant for: API testing only

Using the Security Settings dialog box, you can configure security settings for all operations in a Web service port. To set the security for a specific step within your test, use the **Security Settings** tab in the Properties pane. For details, see "Result Tab (Properties Pane - API Testing)" on page 427.



Security Settings for Port 'FlightsServiceMethods'

Service Details: Web Service

User name:

Password:

Client certificate:

Use default proxy: ☑

Proxy URL:

Proxy user name:

Proxy password:

Keep alive: □

Connection timeout: 0

Manage cookies ☑

Service Details:Web Service

HTTP Transport Level settings, such as username and password for basic authentication, HTTP proxy settings, and Message Level settings. This section also shows the WS-Security including the username token, X.509 certificate signatures, encryption, and so forth.

To access	1. Do one of the following: ▪ Ensure that an API test or component is in focus in the document pane. ▪ In the solution explorer, select an API test or component. 2. Import a Web Service. 3. Select a Web Services port node in the Toolbox pane and select **Security Settings** from the context menu.
Important information	• For details about choosing a security scenario type, see "Security Scenarios Overview" on page 1878. • For examples, see "How to Set Common Web Services Security Scenario Properties" on page 1893.
Relevant tasks	"How to Set Security for a Specific Step" on page 1886

User interface elements are described below:

UI Elements	Description
Service Details	The type of Web service. You can choose from the following types of services: - **Web Service** (standard) - **WCF CustomBinding** - **WCF Federation** - **WCF WSHttpBinding**
<Service security settings>	The settings for the selected service. The available options depending on the type of service selected. - For standard **Web services**, see "Web Service Scenario (Security Settings for Port <Port_Name> Dialog Box)" below - For WCF CustomBinding services, see "WCF Service (Custom Binding) Scenario (Security Settings for Port <Port_Name> Dialog Box" on page 1909 - For WCF Federation services, see "WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 1911 - For WCF WSHttpBinding services, see "WCF Service (WSHttpBinding) Scenario (Security Settings for Port <Port_Name>) Dialog Box)" on page 1913
Advanced	Opens the Advanced Settings dialog box. For details, see the "Advanced Settings Dialog Box" on page 1916. **Note:** Available only for WCF type Web services.
Import	Loads security settings from a previously saved .stss file.
Save	Saves the security scenario settings to an .stss (Service Test Security Scenario) file, for use in other tests. If you are connected to ALM, it saves the file together with the test.

Web Service Scenario (Security Settings for Port <Port_Name> Dialog Box)

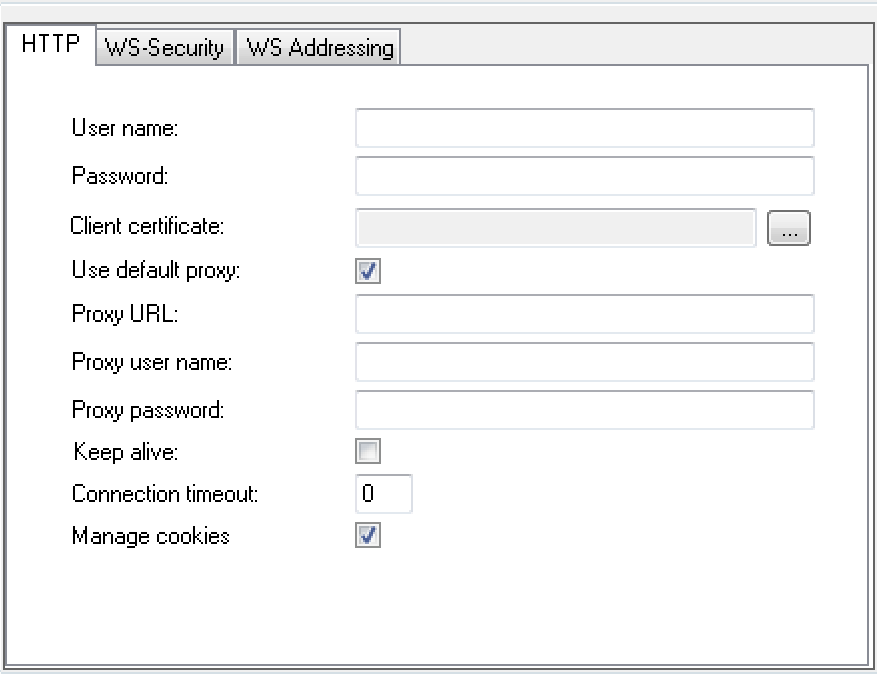
Relevant for: API testing only

When you select the standard Web Service, you set multiple different options in the following tabs:

- "HTTP tab" below
- "WS-Security tab" on the next page
- "WS-Addressing tab" on page 1908

HTTP tab

This tab enables you to set the HTTP transport level settings such as user credentials for sending a message with basic authentication, proxy settings, message-level settings, encryption, and so forth.



User name:	
Password:	
Client certificate:	...
Use default proxy:	☑
Proxy URL:	
Proxy user name:	
Proxy password:	
Keep alive:	□
Connection timeout:	0
Manage cookies	☑

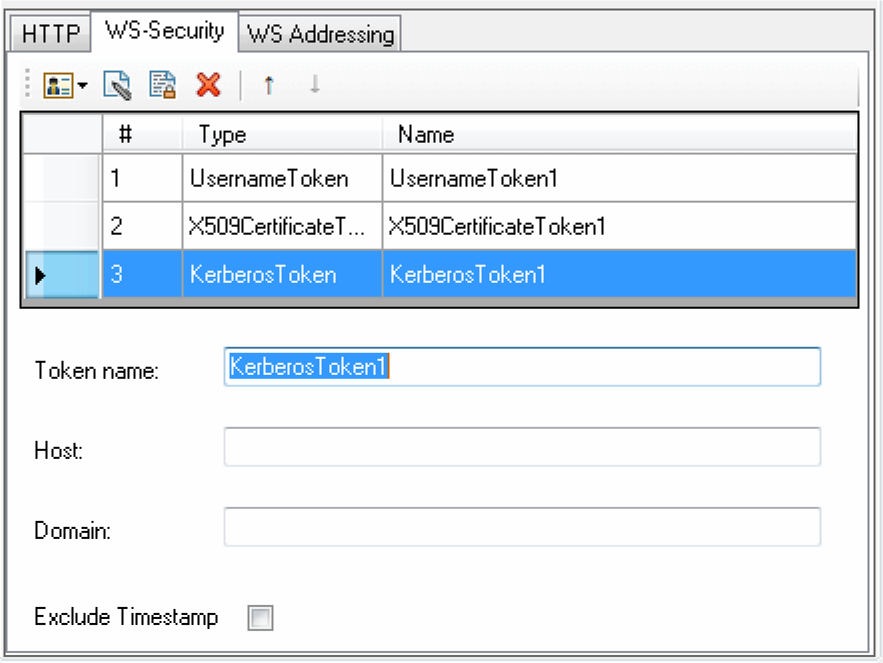
The user interface elements are described below.

UI Elements	Description
User name **Password**	The credentials for HTTP authentication such as basic authentication, digest, or NTLM. **Example** • **User name:** myDomain\myUser • **Password:** myPassword

UI Elements	Description
Client certificate	The client credentials required for client certificate authentication when using two-way SSL scenarios. The **Browse** button opens the "Select Certificate Dialog Box" on page 1923.
Proxy URL	The URL and port of the proxy server through which the message must pass. **Example:**http://myProxy:8888/. To use the default, select **Use default proxy**.
Proxy user name **Proxy password**	The credentials for the proxy server through which the message must pass.
Keep alive	Keeps the connection persistent.
Connection timeout	The time threshold in which to connect through the proxy server or with authentication.
Manage cookies	Enables the writing of cookie information.

WS-Security tab

The tab enables you to add message level security using tokens, message signatures, and encryption.



#	Type	Name
1	UsernameToken	UsernameToken1
2	X509CertificateT...	X509CertificateToken1
3	KerberosToken	KerberosToken1

Token name: KerberosToken1

Host:

Domain:

Exclude Timestamp □

The user interface elements are described below. (Unlabeled elements are shown in angle brackets).

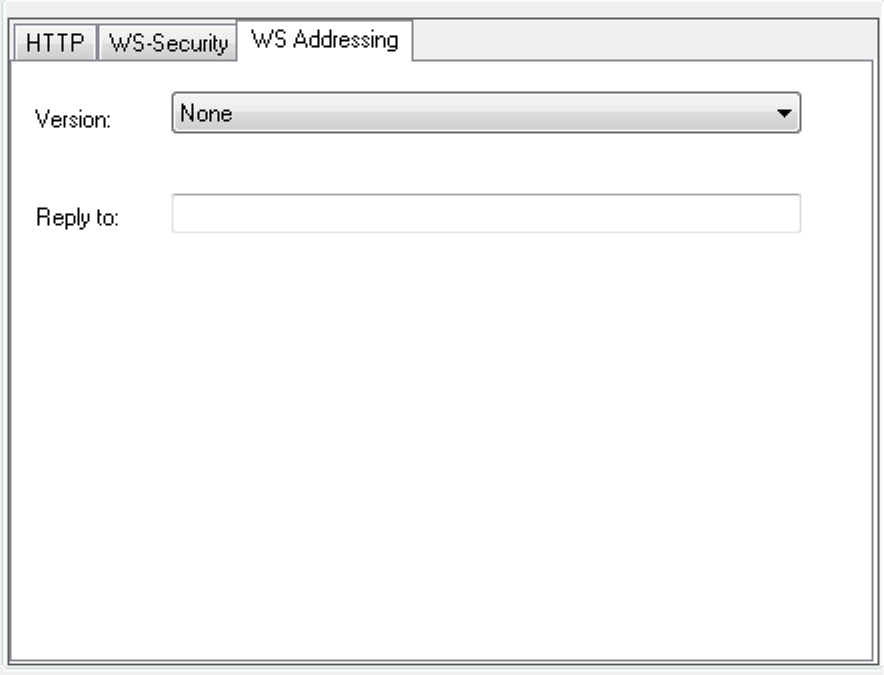
UI Elements	Description
	Security Tokens. Adds a security token to your Web service. You can select one of the following types of tokens: - **User Name**, - **X509** - **Kerberos**, - **Kerberos2** - **SAML**
	Add Message Signature. Adds a signature to the message. This requires a token.
	Add Message Encryption. Adds encryption to the message. This requires a token.
<security element list>	A list of the tokens, message signatures, and encryptions.
<Encryption details pane>	The details of the encryption token. - **Encrypting token.** The token to use for encryption, usually an X.509 type. You can select from a list of all previously created tokens. - **Encrypting type.** Indicates whether to encrypt the whole destination Element or only its Content. - **Key algorithm.** The algorithm to use for the encryption of the session key: RSA15 or RSAOAEP. - **Session algorithm.** The algorithm to use for the encryption of the SOAP message. You can select from a list of common values. - **What to encrypt** ▪ **XPath (optional).** An XPath that indicates the parts of the message to encrypt. If left blank, only the SOAP body is encrypted. ▪ **Token (optional).** The name of the encrypted token. A drop down box provides a list of all added tokens. With most services, this field should be left empty.

UI Elements	Description
<Signature details pane>	The details of the digital signature used to secure the token. • **Signing token.** The token to use for signing, usually an X.509 type. Select from the list of all added tokens. • **Canonicalization algorithm.** A URL for the algorithm to use for canonicalization. A drop down list provides common algorithms. If you are unsure which value to use, keep the default. • **Transform algorithm.** A URL for the Transform algorithm to apply to the message signature. A drop down list provides common algorithms. If you are unsure which value to use, keep the default. • **Inclusive namespaces list.** A list of comma-separated prefixes to be treated as inclusive (optional). • **What to sign.** The SOAP elements to sign: SOAP Body, Timestamp, and WS-Addressing. • **XPath (optional).** An XPath that specifies which parts in the message to sign. If left blank, the elements selected in the **Signature options** field are signed. For example, //*[local-name(.)='Body']. • **Token (optional).** The target token you want to sign. Select from the drop down list of all added tokens. With most services, this field should be left empty.
<Token details pane> - for Kerberos tokens	Token details for **Kerberos** tokens: • **Token name.** A meaningful name for the token. • **Host.** The host name of the server against which you want to authenticate. In most cases, it is the host portion of the service URL. • **Domain.** The Windows domain of the server against which you want to authenticate.

UI Elements	Description
<Token details pane> - for Username tokens	Token details for **Username** tokens. - **Token name.** A meaningful name for the token (you can use the default value). - **Include nonce.** Includes a nonce in the token. - **User name, Password** - **Password type:** Text, Hash, or None. - **Timestamp format:** Full, Created, or None.
<Token details pane> - for X509 Certificate tokens	- **Token name.** A meaningful name for the token. - **Certificate.** The path of the server certificate file. The **Browse** button opens the "Select Certificate Dialog Box" on page 1923. - **Reference type.** Instructs UFT how to reference the security token: BinarySecurity Token or Reference. When the certificate is used for encryption, for example, a service certificate, use Reference. When using it for a signature (for example, a certificate with your private key) select BinarySecurity Token.
<Token details pane> -for SAML tokens	- **<SAML token assertions>.** ▪ **Grid** view, an expandable node listing the assertions in the SAML schema. If you expand the list, you can set the attribute values. A drop down list lets you select a specific assertion. ▪ **Text** view, an XML reference to the token. ▪ **Revert.** Discards all changes made in Text view. - **Load from file.** Enables you to browse to a SAML certificate. - **Certificate.** The path of the certificate file. The **Browse** button opens the "Select Certificate Dialog Box" on page 1923. - **Certificate reference type.** Instructs UFT how to reference the certificate: X509, Data, or RSA.
Exclude Timestamp	Removes the timestamp from the SOAP header before sending the security element to the server.

WS-Addressing tab

The **WS-Addressing** tab indicates whether WS-Addressing is used by the service, and if so, its version number. You can also specify the IP address of the server to which you want the response to be sent.



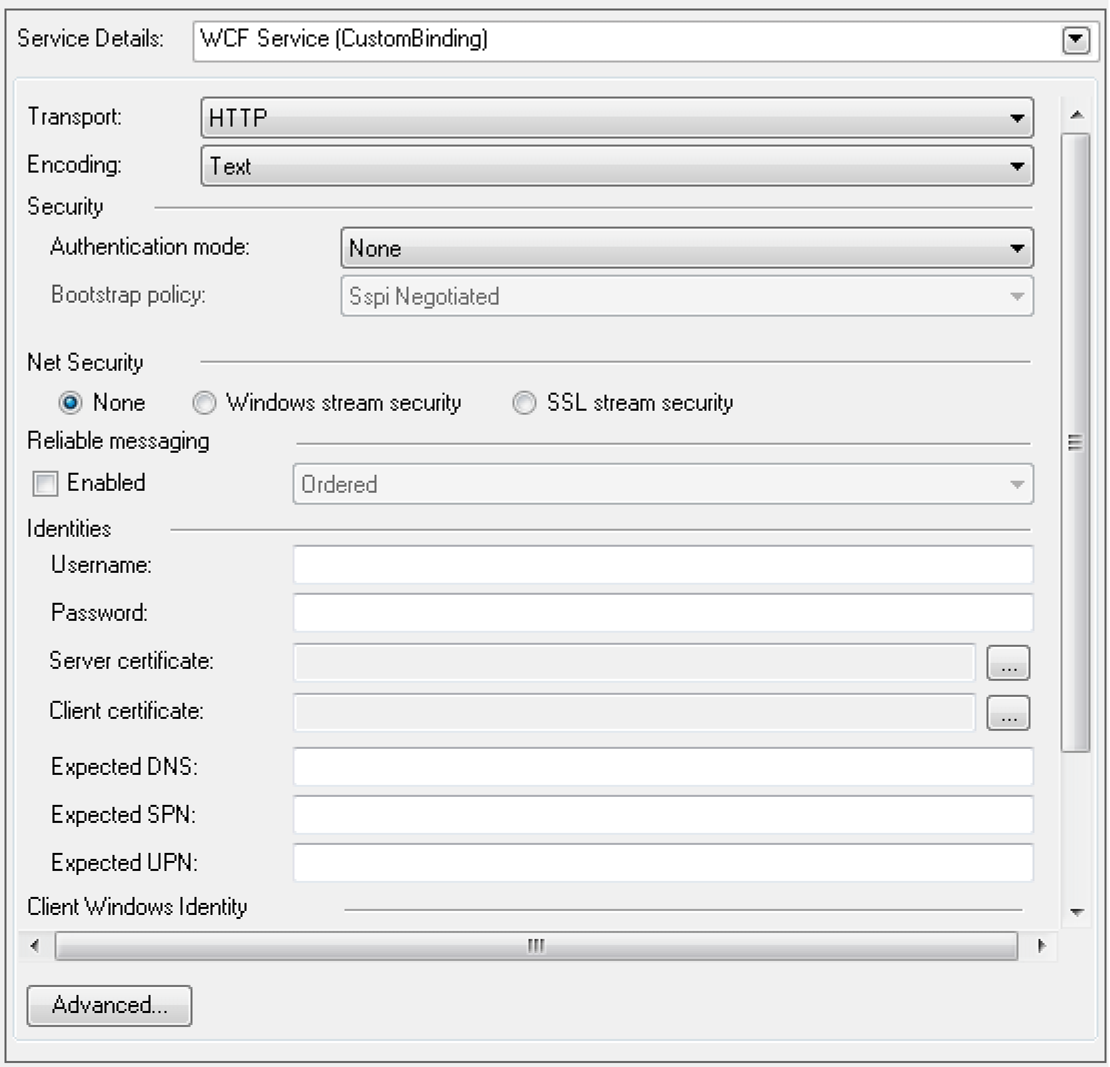
Version: None

Reply to:

WCF Service (Custom Binding) Scenario (Security Settings for Port <Port_Name> Dialog Box

Relevant for: API testing only

Use this scenario to test WCF services which require security or transport configurations. For general details, see "WCF Service (CustomBinding) Scenario Overview" on page 1881.



Service Details: WCF Service (CustomBinding)

Transport: HTTP

Encoding: Text

Security

Authentication mode: None

Bootstrap policy: Sspi Negotiated

Net Security

● None ○ Windows stream security ○ SSL stream security

Reliable messaging

□ Enabled Ordered

Identities

Username:

Password:

Server certificate:

Client certificate:

Expected DNS:

Expected SPN:

Expected UPN:

Client Windows Identity

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

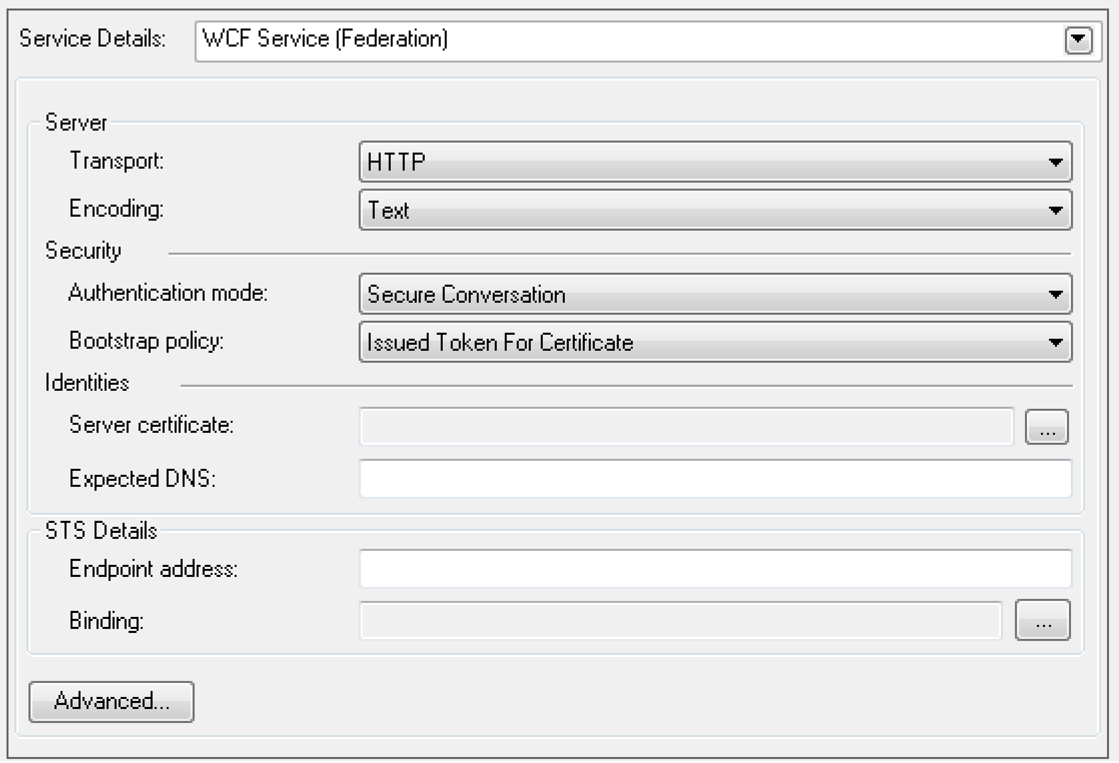
UI Elements	Description
Transport	**Transport type.** The transport types include: • HTTP • HTTPS • TCP • NamedPipe • AutoSecuredHTTP
Encoding	**Encoding.**You can choose from the following encoding types: • Text • MTOM • WCF Binary
Security	• **Authentication mode.** A drop down list of possible modes of authentication, such as AnonymousForCertificate, MutualCertificate, and so forth. • **Bootstrap Policy.** A drop down list of possible bootstrap policies for Secure Conversation authentication., such as SspiNegotiated, UserNameOverTransport, and so forth.
Net Security	The type of stream security: **None, Windows stream security**, or **SSL stream security**.
Reliable Messaging	Enables **Reliable Messaging** in Ordered or Non-ordered format.
Identities	The identity information for the bindings and certificate: • **Username** and **Password** • **Server /Client certificate.** A certificate that provides identity information for the server or client. Use the **Browse** button to open the "Select Certificate Dialog Box". • **Expected DNS**, **SPN**, and **UPN.** The expected identity of the server in terms of its DNS, SPN, or UPN. This can be localhost, an IP address, or a server name.

UI Elements	Description
Client Windows Identity	The identity information for the client windows: - **Current User.** The identity of the user logged onto the machine. - **Custom User.** A user with the following credentials: **Username**, **Password**, and **Domain.**
Advanced	Opens the Advanced Settings dialog box. For details, see "Advanced Settings Dialog Box" on page 1916.

WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)

Relevant for: API testing only

In the **WCF Service (Federation)** scenario, the client authenticates against the STS (Security Token Service) to obtain a token. The client uses the token to authenticate against the application server.



Service Details: WCF Service (Federation)

Server

Transport: HTTP

Encoding: Text

Security

Authentication mode: Secure Conversation

Bootstrap policy: Issued Token For Certificate

Identities

Server certificate:

Expected DNS:

STS Details

Endpoint address:

Binding:

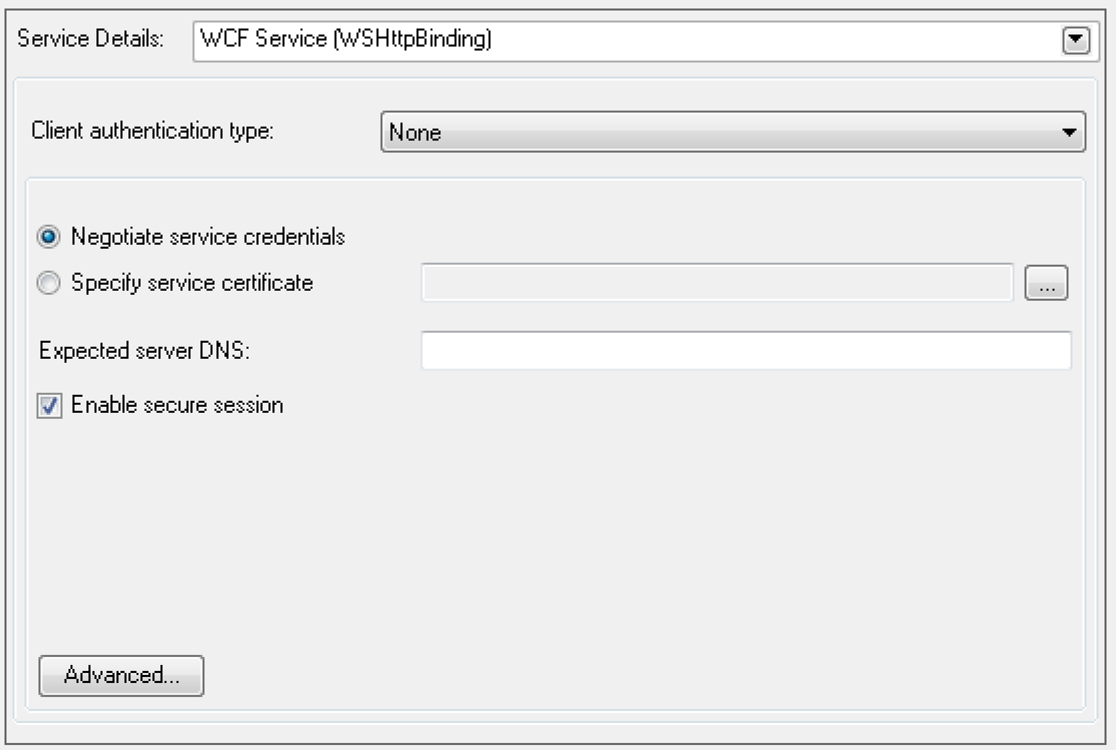
User interface elements are described below (unlabeled elements are shown in angle brackets). For details, see "WCF Service (Federation) Scenario Overview " on page 1882.

UI Elements	Description
Server	• **Transport.** The transport type: HTTP or HTTPS. • **Encoding.** The server's encoding policy: Text or MTOM.
Security	• **Authentication mode.** A drop down list of possible modes of authentication, such as AnonymousForCertificate, MutualCertificate, and so forth. • **Bootstrap Policy.** A drop down list of possible bootstrap policies for Secure Conversation authentication., such as SspiNegotiated, UserNameOverTransport, and so forth.
Identities	The identity information for the bindings and certificate: • **Server certificate.** A certificate that provides identity information for the server. Use the **Browse** button to open the "Select Certificate Dialog Box". • **Expected DNS.** The expected identity of the server in terms of its DNS. This can be localhost, an IP address, or a server name.
STS (Security Token Service) Details	Information about the STS: • **Endpoint address.** The endpoint address of the STS. This can be localhost, an IP address, or a server name. • **Binding.** The scenario which references the binding that contacts the STS.
Advanced	Opens the Advanced Settings dialog box. For details, see "Advanced Settings Dialog Box" on page 1916.

WCF Service (WSHttpBinding) Scenario (Security Settings for Port <Port_Name>) Dialog Box)

Relevant for: API testing only

In the **WCFService (WSHttpBinding)** scenario, you can select from several types of authentication: None, Windows, Certificate, or Username (message protection). For details, see "WCF Services (WSHttpBinding) Scenario Overview " on page 1883.



Service Details: WCF Service (WSHttpBinding)

Client authentication type: None

● Negotiate service credentials

○ Specify service certificate

Expected server DNS:

☑ Enable secure session

User interface elements are described below (unlabeled elements are shown in angle brackets) by the client authentication types:

UI Elements	Description
Client authentication type	Authentication type: - **None**, - **Windows** - **Certificate** - **Username (message protection)** For details, see below.

UI Elements	Description
Advanced	Opens the Advanced Settings dialog box. For details, see "Advanced Settings Dialog Box" on page 1916.

Client Authentication Types:

- "Client Authentication Type — None" below
- "Client Authentication Type — Windows" below
- "Client Authentication Type — Certificate" on the next page
- "Client Authentication Type — Username (message protection)" on the next page

Client Authentication Type — None

UI Elements	Description
Expected server DNS	The expected identity of the server in terms of its DNS. This can be localhost, an IP address, or a server name. It can also be the common name by which the certificate was issued.
Negotiate server credentials	Negotiates the Web Service's certificate with the server. You can also provide the server's DNS information.
Specify service certificate	The location of the service's certificate. If you select this option, the **Negotiate service credentials** option is not relevant. For details, see the "Select Certificate Dialog Box" on page 1923.
Enable secure session	Enables a secure session using no client authentication.

Client Authentication Type — Windows

UI Elements	Description
Client Windows identity	The identity information for the client windows: • **Current User.** The identity of the user logged onto the machine • **Custom User.** A user with the following credentials: **Username**, **Password**, and **Domain**
Enable secure session	Enables a secure session using Windows type authentication.

UI Elements	Description
Expected server identity	The expected server identity method: SPN or UPN.

Client Authentication Type — Certificate

UI Elements	Description
Client certificate	The location of the client certificate. The **Browse** button opens the "Select Certificate Dialog Box".
Enable secure session	Enables a secure session using Certificate type authentication.
Expected server DNS	The expected identity of the server in terms of its DNS. This can be localhost, an IP address, or a server name. It can also be the common name by which the certificate was issued.
Negotiate server credentials	Negotiates the Web Service's certificate with the server. You can also provide the server's DNS information.
Specify service certificate	The location of the service's certificate. If you select this option, the **Negotiate server credentials** option is disabled. For details, see the "Select Certificate Dialog Box" on page 1923.

Client Authentication Type — Username (message protection)

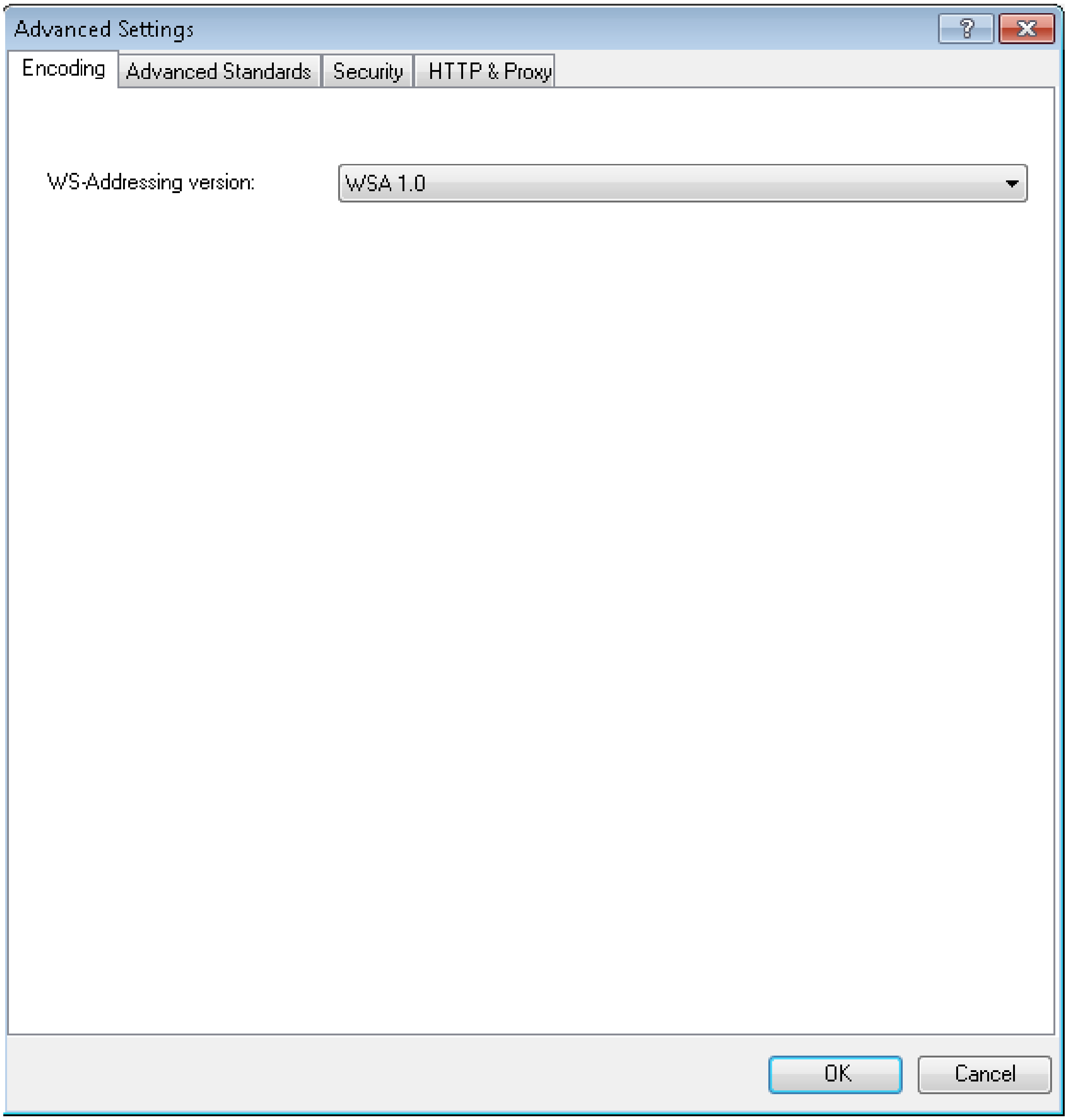
UI Elements	Description
Enable secure session	Enables a secure session using Username type authentication.
Expected server DNS	The expected identity of the server in terms of its DNS. This can be localhost, an IP address, or a server name. It can also be the common name by which the certificate was issued.
Negotiate server credentials	Negotiates the Web Service's certificate with the server. You can also provide the server's DNS information.
Specify service certificate	The location of the service's certificate. If you select this option, the **Negotiate server credentials** option is disabled. For details, see the "Select Certificate Dialog Box" on page 1923.

UI Elements	Description
Username, Password	The authentication credentials of the client.

Advanced Settings Dialog Box

Relevant for: API testing only

This dialog box enables you to customize the security settings for your test.



WS-Addressing version: WSA 1.0

To access	Do the following: 1. Open the "Security Settings for Port <Port_Name> Dialog Box". 2. Select a WCF Service scenario type from the **Service Details** list. 3. Click **Advanced**.
Relevant tasks	- "How to Set Security for a Web Service on the Port Level" on page 1885 - "How to Set Security for a Specific Step" on page 1886 - "How to Customize Security for WCF Type Web Services" on page 1889

This dialog box includes the following tabs:

- "Encoding Tab (Advanced Settings Dialog Box)" below
- "Advanced Standards Tab (Advanced Settings Dialog Box)" on the next page
- "Security Tab (Advanced Settings Dialog Box)" on page 1919
- "HTTP and Proxy Tab (Advanced Settings Dialog Box)" on page 1921

Encoding Tab (Advanced Settings Dialog Box)

Relevant for: API testing only

This tab lets enables you to select the type of encoding to use for the Web service's messages.

User interface elements are described below:

UI Elements	Description
Encoding	The encoding type to use for the messages. You can select any of the following types of encoding: - **Text** - **MTOM** - **WCF Binary**
WS-Addressing version	The version of WS-Addressing for the selected encoding. You can select from the following versions: - **None** - **WSA 1.0** - **WSA 04/08**

Advanced Standards Tab (Advanced Settings Dialog Box)

Relevant for: API testing only

This tab enables you to configure advanced WS- standards, such as Reliable Messaging and the Via address option.

You can configure the following options:

UI Elements	Description
Reliable messaging	Enables reliable messaging for services that implement the **WS-ReliableMessaging** specification. You can use any of the following types of encoding: • **Text** • **MTOM** • **WCF Binary**
Reliable messaging ordered	Indicates whether the reliable messaging session should be ordered.
Reliable messaging version	The version to apply to the messages: WSReliableMessagingFebruary2005 or WSReliableMessaging11.
Specify via address	Sends a message to an intermediate service that submits it to the actual server. This may also apply when you send the message to a debugging proxy. This corresponds to the **WCF clientVia** behavior. This is useful to separate the physical address to which the message is actually sent, from the logical address for which the message is intended.
Via address	The logical address to which to send the message. It may be the physical of the final server or any name. It appears in the SOAP message as follows: `<wsa:Action>http://myLogicalAddress<wsa:Action>` The logical address is retrieved from the user interface. By default, it is the address specified in the WSDL. You can override this address using this field.

Security Tab (Advanced Settings Dialog Box)

Relevant for: API testing only

The Advanced **Security** settings correspond to **WS-Security** specifications. You can use the following settings:

UI Element	Description
Enable secure session	Establish a security context using the WS-SecureConversation standard.
Negotiate service credentials	Allow WCF proprietary negotiations to negotiate the service's security.
Default algorithm suite	The algorithm to use for symmetric/asymmetric encryption. The algorithm drop down list gets its values from the **SecurityAlgorithmSuite** configuration in WCF. **Default:** Basic256
Protection level	Indicates whether the SOAP Body be encrypted/signed. The possible values are: None, Sign, and Encrypt And Sign (default). **Default:** Encrypt And Sign
Message protection order	The order for signing and encrypting. Choose from: • Sign Before Encrypt • Sign Before Encrypt-And Encrypt Signature • Encrypt Before Sign
Message security version	The WS-Security security version. You can also indicate whether to **Require derived keys** for the message.
Require derived keys	Indicates whether to require derived keys.

UI Element	Description
Security header layout	The layout for the message header: - **Strict** - **Lax** - **Lax Timestamp First** - **Lax Timestamp Last**
Key entropy mode	The entropy mode for the security key. The possible values are: Client Entropy, Security Entropy, and Combined Entropy. **Default:** Combined Entropy
Require security context cancellation	Indicates whether to require the cancellation of the security context. If you disable this option, stateful security tokens will be used in the **WS-SecureConversation** session, if they are enabled.
Include timestamp	Includes a timestamp in the header.
Allow serialized signing token on reply	Enables the reply to send a serialized signing token.
Require signature confirmation	Instructs the server to send a signature confirmation in the response.
X509 inclusion mode	When to include the X.509 certificate: - Always to Recipient - Never - Once - Always To Initiator **Note:** This and the next three options only apply when using an X.509 certificate.

UI Element	Description
X509 reference style	How to reference the certificate: - Internal - External
X509 require derived keys	Indicates whether X.509 certificates should require derived keys.
X509 key identifier clause type	The type of clause used to identify the X.509 key. - Any - Thumbprint - Issuer Serial - Subject Key Identifier - Raw Data Key Identifier

HTTP and Proxy Tab (Advanced Settings Dialog Box)

Relevant for: API testing only

This tab lets you set the HTTP transfer and Proxy information for your test.

The following options are available:

UI Element	Description
Transfer mode	The transfer method for requests/responses. The possible values include: - **Buffered** - **Streamed** - **Streamed Request** - **Streamed Response**
Max response size (KB)	The maximum size of the response before being concatenated. **Default:** 65 KB
Send Timeout	The time to wait (in seconds) for response on the transfer.
Allow cookies	Indicates whether to enable or disable cookies.

UI Element	Description
Transfer mode	The transfer method for requests/responses. The possible values include: - **Buffered** - **Streamed** - **Streamed Request** - **Streamed Response**
Keep-Alive enabled	Indicates whether to enable or disable keep-alive connections.
Authentication scheme	The HTTP authentication method: - **None** - **Digest** - **Negotiate** - **NTLM** - **Integrated Windows Authentication** - **Basic** - **Anonymous**
Realm	The realm of the authentication scheme in the form of a URL.
Require client certificate	Indicates whether to require a certificate for SSL transport.
Use default web proxy	Indicates whether to use machine's default proxy settings.
Bypass proxy on local	Indicates whether to ignore the proxy when the service is on the local machine.
Proxy address	The URL of the proxy server.
Proxy authentication scheme	HTTP authentication method on the proxy server: - **Digest** - **Negotiate** - **NTLM** - **Basic** - **Anonymous**

Select Certificate Dialog Box

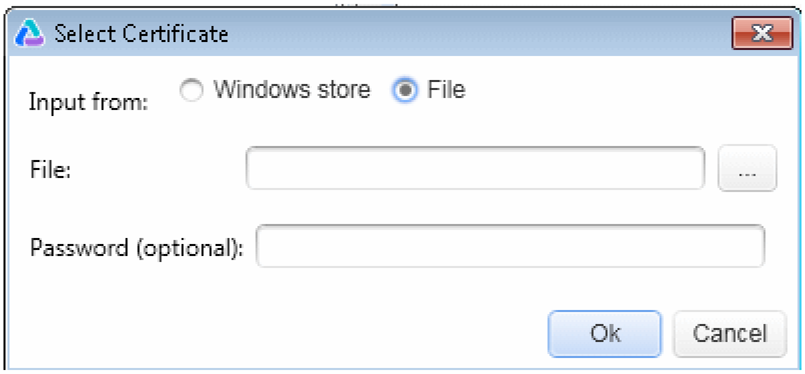
Relevant for: API testing only

This dialog box enables you to search and locate a certificate from a file or Windows store.

To access	Do the following: 1. Open the "Security Settings for Port <Port_Name> Dialog Box". 2. Select a WCF Service scenario type from the **Service Details** list. 3. Click the **Browse** button adjacent to the **Server Certificate** box.
Relevant tasks	"How to Customize Security for WCF Type Web Services" on page 1889

The available options differ depending on whether you are selecting your certificate from a file or from the Windows store:

Select Certificate from File

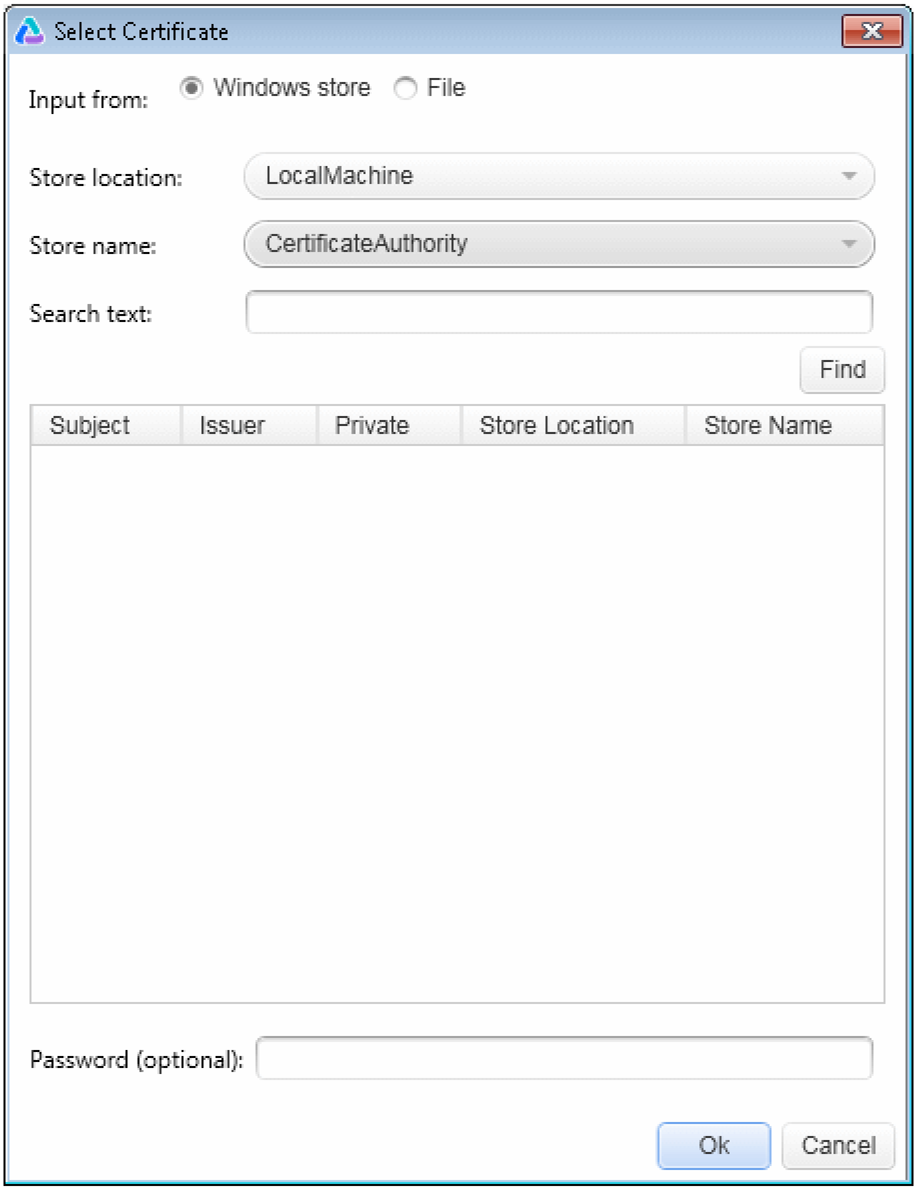


Input from: ○ Windows store ● File

User interface elements are described below:

UI Elements	Description
...	**Browse.** Enables you to locate the certificate file with a .cer or .pfx extension.
File	The complete path of the certificate file.
Password (optional)	The password required to access the certificate.

Select Certificate from Windows Store



User interface elements are described below:

UI Elements	Description
Store location	The store location, for example **Current User**.
Store name	The store name, for example, **AuthRoot**.
Search text	The text to match in the certificate name. If left blank, the **Find** action retrieves all available certificates.

UI Elements	Description
<certificate list>	A list of the certificates in the Windows store sorted by **Subject**, **Issuer**, **Private**, **Store Location**, and **Store Name**.
Password (optional)	The password required to access the certificate.

Troubleshooting and Limitations - Web Service Security

Relevant for: API testing only

This section describes troubleshooting and limitations for working with Web services security.

- Authentication and proxy security are not supported for Web Services imported from a UDDI.
- For Web Service configured with WCF settings: Configuring different security settings for operations residing on the same port is not supported.
- For Web Service configured with WCF settings, some of the user event handlers (such as the **AfterProcessRequestSecurity**, **BeforeProcessResponseSecurity**, **OnSendRequest**, and **OnReceiveResponse** events) will not be invoked.
- When testing Web Services that require message-level security, the Web Service security scenario only supports SOAP version 1.1. For SOAP 1.2 use a WCF type scenario.
- When using a SAML security token for Web services security, user-provided content may contain creation and expiration timestamps. To extend the life of the test, we recommend that you hard-code an expiration date in the distant future. In this is not possible, change the timestamp by implementing the **OnBeforeApplyProtocolSettings** event.
- When using a SAML security token for Web services security, if you edit the values in Grid mode, they may not be updated in UFT.

  Workaround: To update the values, switch to **Text** mode and save the test.
- Web Service steps are not supported when using a SAML token with a certificate from the file system.

  Workaround: Install the certificate to the Windows store and select the certificate from the store.
- When working with Federation type scenarios that use STS (Security Token Service), you cannot change the SOAP version.
- If you are using SOAP version 1.2:
  - You can choose only UserName or X509 tokens when configuring the message level security.
  - When configuring the canonicalization algorithm and the transform algorithm for the message signature, you cannot use the following format: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform.

Chapter 61: Asynchronous Service Calls

Relevant for: API testing only

This chapter includes:

Concepts

Asynchronous Services

Relevant for: API testing only

You can use UFT to emulate asynchronous services. Asynchronous services can be Web Services, REST services, HTTP requests, JMS/MQ-based services, and so forth.

In synchronous messaging, the replay engine blocks step execution until the server responds. The client sends a request and receives a response immediately, using the same connection. During the waiting time, the replay engine is blocked and does not perform any other activity. If the timeout was reached without a response from the server, the client returns an error.

In asynchronous mode, the replay engine executes the step without waiting for server's response from previous requests.

Wait Steps

When sending asynchronous calls with HTTP or HTTPS, you use a **Wait** step to instruct the test to wait for the response of earlier asynchronous requests before continuing with its execution.

You do not have to place the **Wait** step directly after the Receive step. The test can proceed with other steps, but the **Wait** step will instruct the test to wait for a response before ending the test, or before continuing to execute any steps that follow the **Wait** step.

Checkpoints

When sending a request to a server, you use checkpoints to verify the **Output** property values.

When getting a request from a server, as in most of the asynchronous patterns, you use checkpoints to verify the **Input** properties.

UFT provides a solution for the following asynchronous patterns:

- "WS-Addressing" below
- "HTTP Receiver" on the next page
- "Web Service Publish Subscribe" on the next page
- "Web Service Solicit Response" on the next page
- "Dual WSDL Files" on page 1930

WS-Addressing

WS-Addressing is a specification that enables Web services to communicate addressing information. You can instruct the server to respond to any location, and not necessarily to the machine that issued the request. To do this, you use the WS-Addressing **replyTo** attribute.

In this implementation, UFT pauses the test and uses a listener mechanism to verify that the response arrived at the specified address. After the listener acknowledges that the server responded to the address or if it reaches the timeout, the test resumes. Upon the completion of the test, you can validate the response with the standard API testing checkpoints.

For user interface details, see the "Asynchronous Tab (Properties Pane - API Testing)" on page 403.

HTTP Receiver

In the **HTTP Receiver** pattern, the **server** sends an HTTP request to the client, reversing the typical roles of the client and server.

This is useful, for example, if you want to test a service which publishes information over HTTP to a client. You define a receiver, which waits for a request from the server, sent over HTTP.

After a trigger, the receiver captures the request. The trigger can be an HTTP client request, a call to a Web service, an email, or any other event that will trigger the server. If there are inner steps, the receiver waits for them to finish and only then is the receiver activity considered complete.

Using the API testing interface, you can insert the necessary logic and validate the checkpoints in the captured request.

The response from the receiver should wait for the inner steps to complete and link to them.

The completion event name fired for the receiver should only be fired AFTER the inner steps are done.

For details about the properties, see the "HTTP Receiver Activity" on page 1687.

Web Service Publish Subscribe

In the **Web Service Publish Subscribe** pattern, the **server** sends an HTTP request to the client, reversing the typical roles of the client and server. it is similar to the HTTP Receiver, except that the request is sent to the client through a Web Service call instead of exclusively via HTTP.

Using UFT, you test the publishing of messages to the client. You set up a receiver, which waits for a server request, sent from the server as a Web service call.

After a trigger, the receiver captures the request. The trigger can be an HTTP client request, a call to a Web service, an email, or any other event that will trigger the server.

Using the API testing interface, you can validate the response with standard API testing checkpoints.

Web Service Solicit Response

The **Web Service Solicit Response** pattern is a variation of the **Web Service Publish Subscribe** pattern. It enables you test a a service which publishes information through a Web Service to a client.

In this pattern, however, the client is expected to send a response to the server request. The response can be a simple acknowledgment or a full SOAP message.

You set up a receiver activity, which waits for a server request. This server request is sent from the server as a Web service call. The receiver then sends a client response back to the server.

After the trigger, the receiver captures the request. The trigger can be an HTTP client request, a call to a Web service, an email, or any other event that will trigger the server.

Using the API testing interface, you can validate the response with standard API testing checkpoints.

Dual WSDL Files

The Dual WSDL technique is a standard request-response pattern. In this pattern, however, the client request is defined by one WSDL, and the server response is defined by another WSDL.

You implement this scenario in two stages:

- Import the Request WSDL file, using the **Import WSDL from** import command.
- Import the Response WSDL file, using the **Import WSDL from** import command, enabling the **Import as Server Response** option. For details, see the "Import/Update WSDL from URL or UDDI Dialog Box" on page 1766 or "Select WSDL Dialog Box" on page 1764.

For task details, see "How to Create an API Test for an Asynchronous Web Service" on the next page.

Tasks

How to Create an API Test for an Asynchronous Web Service

Relevant for: API testing only

This task describes how to create an API test for testing an asynchronous Web service.

For an overview of the asynchronous patterns supported by UFT, see "Asynchronous Services" on page 1928.

The following section describes the following tasks:

- "Create a test for WS-Addressing" below
- "Create a test for HTTP Receiver" below
- "Create a test for a Web service publish subscribe pattern" on the next page
- "Create a test for Dual WSDL Files" on page 1933

Create a test for WS-Addressing

To use WS-Addressing for a Web Service call:

1. Import a Web Service using **Import WSDL > Import WSDL from URL or UDDI** and drag an operation onto the canvas. For details, see "How to Import a WSDL-Based Web Service" on page 1744.
2. Open the Asynchronous tab in the Properties pane and select **This is an asynchronous call** box.
3. Specify a value for the **Listen for response on** property. This is the port to which you expect the server to respond.
4. Open the Security tab in the Properties pane. Select the **WS Addressing** tab.
5. Select a WS-Addressing **Version** and provide and a URL and port (same port as defined for the **Listen for response on** property) in the **Reply to** box, to indicate the destination of the server response.
6. Run the test and perform checkpoint validations as you would with any step.

For more details, see the "Asynchronous Tab (Properties Pane - API Testing)" on page 403.

Create a test for HTTP Receiver

To create a test for an HTTP Receiver in which the client receives the response:

1. Drag a **Network > HTTP Receiver** step onto the canvas.

   a. Make sure you are logged in as an administrator. Administrator privileges are required to run **HTTP Receiver** steps.

   b. In the **General** tab , specify the **General** properties. For details, see the "HTTP Receiver Activity" on page 1687.

   c. In the HTTP Receiver tab , set the **HTTP Receiver** properties. For details, see the "HTTP Receiver Tab (Properties Pane - API Testing)" on page 416.

   d. Set a filter for the HTTP message . For details, see "Filter Settings Tab (Properties Pane - API Testing)" on page 412.

2. Drag a **Flow Control > Wait** step onto the canvas. Specify timeout information and one or more completion events. You can link to the completion event from a prior **HTTP Receiver** step.

3. If required, drag additional activities from the Toolbox pane into the **HTTP Receiver** flow.

4. Run the test and apply the trigger for the server. For details, see the "HTTP Receiver" on page 1929.

> **Tip:** If your test needs to listen to more than one message, receiver steps (such as **HTTP Receiver** or Web Service calls set up as receivers) can be data driven and placed inside a loop. The placement of the **Wait** step—inside or outside of the loop—depends on whether the send order matters:
>
> - If the messages to be sent to the receiver are expected in a specific order, you must place the **Wait** step inside the receiver step's frame. All steps that are contained within the receiver can be data driven using this loop.
>
> - If however, the messages are expected in a random order, place the **Wait** step outside the receiver step. Steps that are contained within the receiver should not be data driven using the same loop as the receiver step and should not link to other steps outside the receiver.

Create a test for a Web service publish subscribe pattern

To create a test to check that messages are properly published to the client:

1. Import a WSDL as a server by enabling the **Import as server** option. For details, see "How to Import a WSDL-Based Web Service" on page 1744. This Web Service should have a receiver type pattern. Drag an operation onto the canvas.

2. Set the input or output properties . For details, see the "Parameters/Checkpoints Tab (Properties Pane - API Testing)" on page 417.

3. Drag a **Flow Control > Wait** step onto the canvas. Specify timeout information and one or more completion events. You can link to the completion event from a prior **HTTP Receiver** step.

4. Run the test and activate the trigger for the server. For details, see "Web Service Publish Subscribe" on page 1929.

Create a test for Dual WSDL Files

To create a test for a dual WSDL pattern, using one WSDL for the request and another for the response:

1. Import the request WSDL file using the standard import command, **Import WSDL from**. The imported operations can send client requests and receive server responses.

2. Import the response WSDL file as a server by enabling the **Import as Server Response** option in the "Select WSDL Dialog Box" or the "Import/Update WSDL from URL or UDDI Dialog Box" (for URL and UDDI imports, click **Advanced Settings** to show the option). The imported operations first receive server requests and then send client responses.

3. Drag a Web service operation with the request onto the canvas.

4. Drag a Web service operation with the response onto the canvas, after the request step.

For details, see "Dual WSDL Files" on page 1930.

Troubleshooting and Limitations - Asynchronous Testing

Relevant for: API testing only

This section describes troubleshooting and limitations for creating asynchronous tests.

For a Web service imported as a server response:

- When enabling the SSL option, UFT temporarily binds the SSL certificate to the specified port on a system, **http.sys**, level. If you end the UFT.exe process from the Task Manager during the listening stage after the binding was added, the binding will not be automatically removed from the system.

  Workaround: Remove the binding manually using a utility such as httpcfg.exe or netsh.exe.

- When working with SSL, certificates from a file are not supported. If you move the test to another machine, the certificate will not be available.

  Workaround: Add the certificate to the local machine store before running the test. If desired, remove it after you finish working with the test.

Chapter 62: Writing Event Handlers for API Test Steps

Relevant for: API testing only

This chapter includes:

Concepts

Event Based Coding - Overview

Relevant for: API testing only

When testing your application's API, you can use event handlers to change or extend how you test your application's process. An **event handler** is a specific occurrence of a defined code process triggered at a specific point in the overall test flow.

When you run a test of your application's API (or run the application itself), each business process is executed as defined in your application's code. These business processes are represented in your test by the test steps you create. However, events (when running the application's code) and event handlers (when running the test) are used at specific places in the application execution or the test run. For example, if you are testing an application using a Web service, the core part of your test is the Web service call processes in which data is sent to and from the Web service. Events and event handlers can be added before, after, and at other points in the test flow, such as a special event handler that runs after the application compiles the Web service call response, sets security for the Web service response data, or adds attachments to the Web service call response.

Event handlers are designed to be run at a specific point in the application/test workflow. As a result, the objects, methods, and properties available in a given event handler are limited to the context of where the event occurs in the application or test workflow. For example, when you are working in an event handler that occurs before an application process/test step, you cannot access the process's/test step's output properties, as these properties are in a part of the application/test that has not yet run.

Example

You have an application based on a Web service, in which the following steps occur:

1. The application generates the Web service request.
2. The application sets the security for the Web service call request.
3. The application adds the attachments for the Web service request.
4. The application sends the request to the Web service.
5. The application receives the responses from the Web service.
6. The application reads the response and the attachments from the response.

Using event handlers, you can code an event handler in your test for any of these steps. For example, if you want to set the request properties of one step based on the response properties of a previous step, you can create code in an event handler called **BeforeExecuteStep**, and enter the property values you would like your test to use. When writing the code for this event handler, the available properties/methods for your code are limited to the objects contained in the context of the step's flow (the output/response properties available in the previous step and the input/request properties of the current step) and the event handler (which takes place immediately after the step).

You could also add another event handler that builds the attachment data into an XML document to attach to the Web service request. Likewise, you could add numerous other event handlers that supplement the core processes that your application performs.

However, if you use an event handler out of the context in which it is designed, the properties/methods of the step with which the event should run are not accessible. Thus, if you try to code an event handler for one of the response steps above after a step involved in making the Web service request (such as generating the Web service request), the properties of the response step - while accessible - are null in the response context, and your test run gets an exception when running this particular event handler code.

For details about how UFT uses event handlers, see "Writing Code for API Test Events - Overview" on the next page.

Writing Code for API Test Events - Overview

Relevant for: API testing only

Each step has predetermined event handlers which are run at specific points in the test execution. Within these event handlers, you can add additional code (above and beyond the test step's regular execution flow) which enables you to define properties, parameters, or variables and additional processes that help to facilitate your test flow. For most test steps, there are three standard event handlers which run before the test step, after the test step, and as a checkpoint for the test step. For Web service and SOAP request steps, there are additional event handlers that mimic the process of a Web service call. For details on the event handler structure, see "API Event Coding - Event Structure Overview" on page 1978.

Examples

Case 1

Before entering customer information into a flight booking service, your application must connect to a database located locally on your computer (which mimics the application connecting to a local database). However, using the existing UI framework of the UFT API test, you cannot connect to the database. Using an event handler designed to run before the test step, you can write code that accesses your database and imports the information to your test to be entered into the flight booking service.

Case 2

After receiving a response from your Web service (in XML format), you need to extract certain information from the response file to use as the input for another test step. You can write an event handler to run after the test step which can read and extract the information from this XML file.

Event handlers should be used to extend the behavior of existing test steps, instead of providing all the property/parameter values for test steps. It is not recommended to use custom code to set the property/parameter values of your steps and execute the steps, but instead to use the grid in the Input/Properties tab in the Properties pane to set these values.

Note: This functionality is in contrast to the manner GUI testing uses coding. GUI tests and components enable you to write the entire test or action flow using code. However, an API event handler or custom code activity is only a portion of the larger test flow.

It is recommended to have experience and/or knowledge in writing code before attempting to write event handlers for your tests. All API test events use the C# language and syntax, even if your application uses a different language. For details on C#, see the Microsoft C# Reference.

Note: Any add-ins you choose in the Add-in Manager when opening UFT do not affect API tests or event/custom coding.

Writing Events for API Tests - Use-case Scenarios

Relevant for: API testing only

The following scenarios describe use-cases on how to use event coding in the context of a test of a realistic application. Each of the scenarios describes the general context in which the application is used, the workflow of the application, and a test created for the application. The steps in the test include possible input and output properties for each step, and also describe how event coding can be used to enhance various test steps.

You can view a use-case scenario for the following types of applications:

- "Web Service application" below
- "REST Service application" on page 1942
- "Standard application" on page 1945

Web Service application

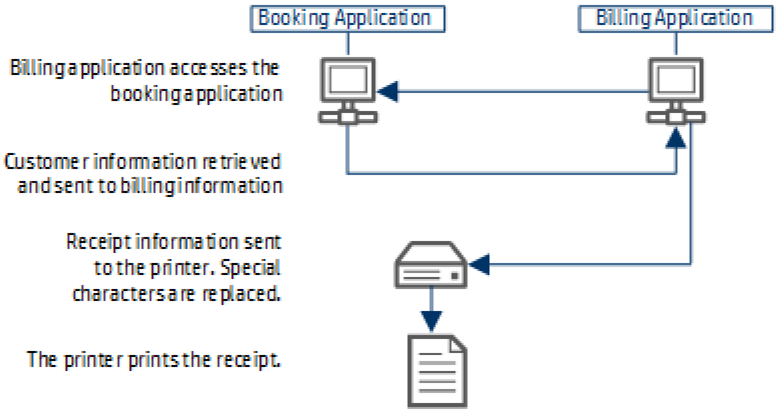
Scenario

A hotel has an application that deals with customer room charges and prints receipts for customers. The application takes the customer’s room number, compiles customer charges of the customer into a bill, charges the customer’s credit card, and prints a receipt. This receipt must display the customer’s name, including any special characters in their name. However, the printer for the receipt can only print English characters.

The billing application is based upon a Web service which works on a cloud-based application and database.

Application Flow

The billing application has the following workflow:


Booking Application
Billing Application
Billing application accesses the booking application
Customer information retrieved and sent to billing information
Receipt information sent to the printer. Special characters are replaced.
The printer prints the receipt.

1. The billing application accesses the hotel booking application where customer information is stored. The hotel billing application retrieves the customer's name, credit card number, and

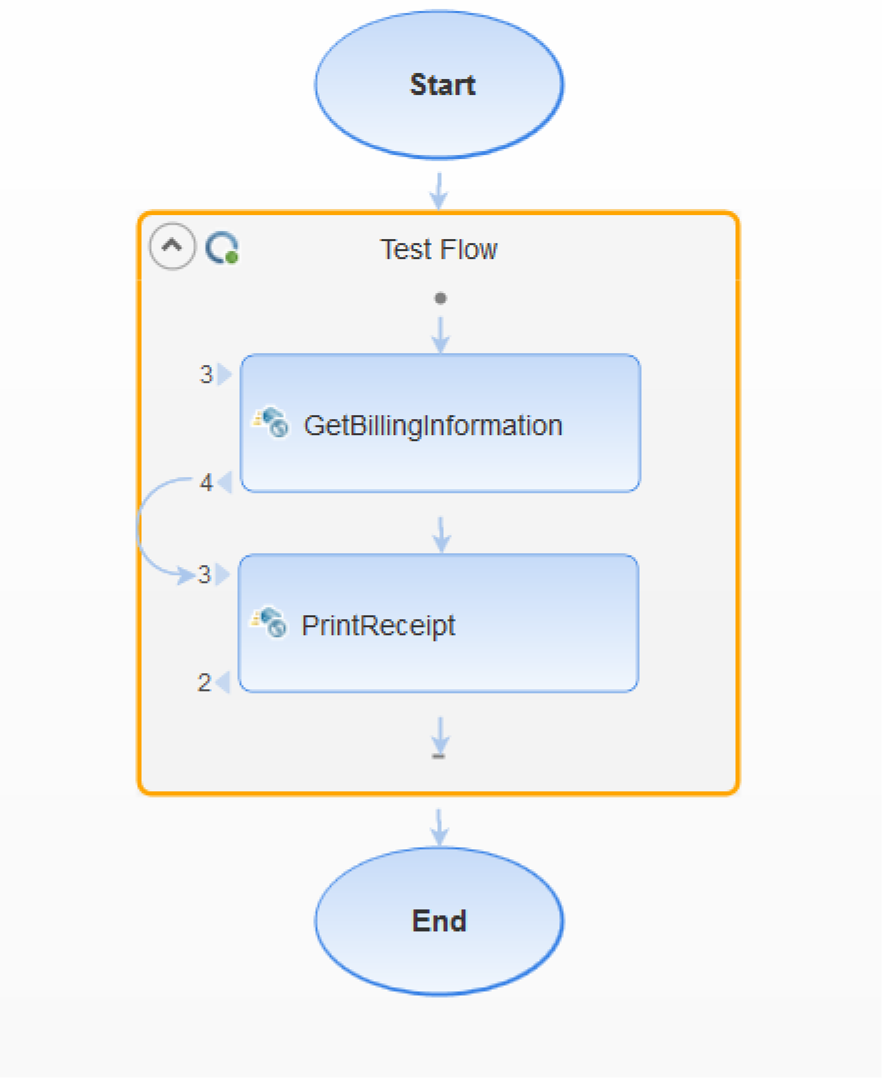
total charges.

2. The billing application sends the information for the bill to the receipt printer.

3. The receipt is printed.

Test Setup

You create the following test for the application, including a number of events:



1. GetBillingInformation step

   This step accesses the hotel booking application in order to retrieve the customer data, including the customer name, total charges, and customer credit card number.

   Properties:

Input Properties	Web service address for the hotel booking application.
Output Properties	- *Customer Name* - *Customer credit card number* These properties are contained in a description element in the Web service response. - *Price*
Checkpoints	Checkpoint to check whether the connection to the hotel booking application succeeded.

2. PrintReceipt step

   This step takes the customer information (from the description element of the GetBillingInformation step) and prints a receipt for the customer.

   Properties:

Input Properties	*Description.* This property is linked to the data source imported in the first step. Price. The total price to charge to the customer.
Output Properties	Response file containing the receipt from the
Checkpoints	None

Events:

For this step, you must create an event handler for the *BeforeExecuteStepEvent* event. This event handler searches the description element for non-English characters and replaces them with the correct characters:

```
 /// <summary>
        /// Handler for the StServiceCallActivity10 Activity’s BeforeExecuteStepEvent event.
        /// </summary>
        /// <param name=\"sender\">The activity object that raised the BeforeExecuteStepEvent
event.</param>
        /// <param name=\"args\">The event arguments passed to the activity.</param>
        /// Use this.StServiceCallActivity10 to access the StServiceCallActivity10 Activity's conte
xt, including input and output properties.
        public void StServiceCallActivity10_OnBeforeExecuteStepEvent(object sender, STActivi
tyBaseEventArgs args)
        {
```

```
        //Get the description text
        XmlNamespaceManager nsmgr = new XmlNamespaceManager(this.StServiceCallAct
ivity10.InputEnvelope.NameTable);
        nsmgr.AddNamespace("a", @"http://schemas.datacontract.org/2004/07/CustomerBil
lingService");
        var OriginalText = this.StServiceCallActivity10.InputEnvelope.SelectSingleNode("//a:
Description", nsmgr).InnerText;

        //In case there are non-English characters in the description, convert them to English
ones
        var ConvertedText = ConvertSpecialCharactersToEnglishCharacters(OriginalText);

        //Update the description with the converted text
        this.StServiceCallActivity10.InputEnvelope.SelectSingleNode("//a:Description", nsmg
r).InnerText = ConvertedText;
    }
```

REST Service application

Scenario

Note: This scenario is based on the Flights API application included with the UFT installation.

You have a flight booking application built using REST services. Your flight application retrieves the flights from the service, creates a flight order, and then reserves the customer order. The service then creates a flight order and total price, which is forwarded to the customer.

Application Flow

The billing application has the following workflow:

1. The flight application accesses the flights database through an HTTP connection, stored externally.

2. The flight application retrieves the flights matching the specified parameters. You can search for flights using any of the following criteria:

   - Airlines

   - Arrival City

   - Arrival Time at destination

   - Departure City

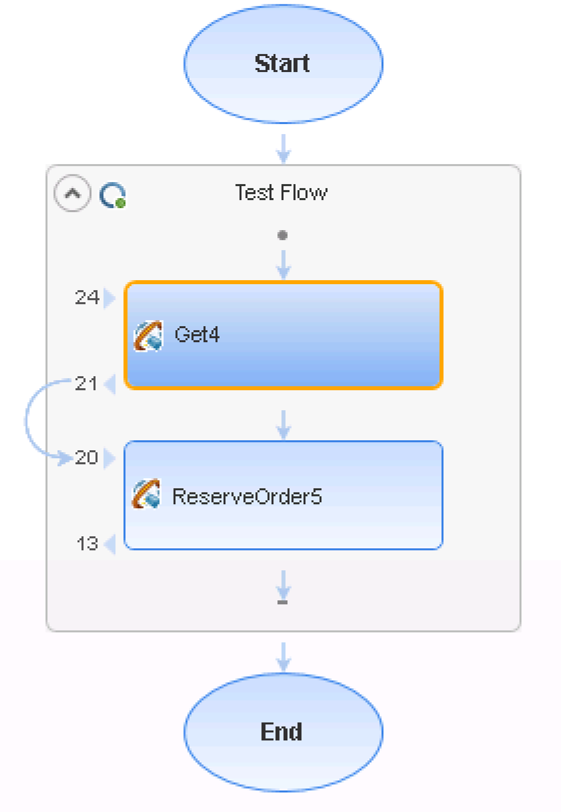
   - Departure Time from departure point

3. The flight application finds a specific flight, using the output from the flight retrieval, creates a flight order, and reserves the customer's place on the flight.

4. The flight application provides the customer with a flight order number and a price.

Test Setup

You create the following test for the application, including a number of events:



1. *Get* step

   In this step, the flight application accesses the flights database, and retrieves the flights that match the customer preferences.

Properties:

Input Properties	- *Airlines.* - *ArrivalCity* - *ArrivalTime* - *DepartureCity* - *DepartureTime* - *FlightNumber*
Output Properties	- *Class* - *DepartureDate* - *FlightNumber* **Note:** The output properties are defined by importing a ResponseXML file, which details the required response parameters.
Checkpoints	None

2. ReserveOrder step

   In this step, the flight application creates a flight order based on the specified output from the Get step, and reserves the customer's place on the flight.

   As part of this step, the test needs to add a checkpoint ensuring that the flight number meets standards of being greater than 999 and less than 100000.

Properties:

Input Properties	■ *Class.* This property is linked to the *Class* output property of the Get step. ■ *CustomerName* ■ *DepartureDate.* This property is linked to the *DepartureDate* output parameter from the Get step. ■ *FlightNumber.* This property is linked to the *FlightNumber* output property from the Get step. ■ *NumberofTickets*
Output Properties	■ *OrderNumber* ■ *TotalPrice*
Checkpoints	A checkpoint ensuring that the flight number meets specified parameters. This checkpoint is added with an event handler.

Events:

For this test step, you add two additional events:

- A CodeCheckpointEvent event. For this event, the code checks that the flight number is greater than 999 and less than 100000. In the event handler, you also add code to stop the test if the checkpoint fails.
- An AfterExecuteStepEvent event. In this event, you add code to save the response XML document as an attachment for the customer to receive.

Standard application

Scenario

You have an application which delivers a response received from a Web-based application. The application creates a file in which to save the response's data, and then extracts the relevant data and adds it to the created file.

Application Flow

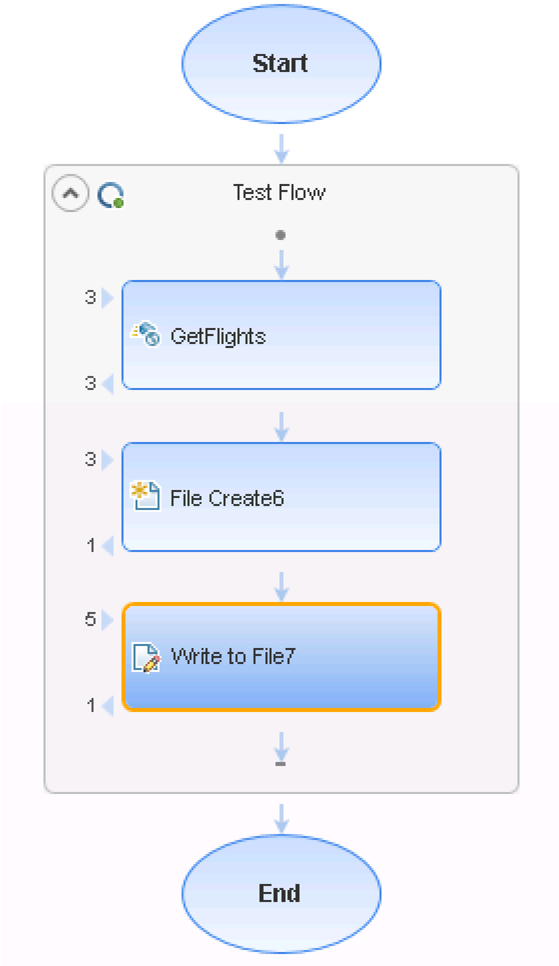
The application has the following workflow:

1. The application receives the response from the Web-based application.
2. The application creates the file for the response data.

3. The application extracts the necessary content from the response, and adds this to the created file.

Test Setup

To test the application, you create the following test:



1. A Web service step.

   This step details the response expected from the Web-based application.

2. A File Cretate step.

   This step creates a file in the specified directory in which the extracted data will be written.

Properties:

Input Properties	- *Folder Path* This property is linked to the *Class* output property of the Get step. - *File Name* Both of these properties are entered manually in the test.
Output Properties	None
Checkpoints	None

3. A Write To File step.

In this step, you extract the data from the Web service response, and then add the data to the file created in the CreateFile step.

Properties:

Input Properties	- *File Path.* This property is linked to the *Folder Path* property of the CreateFile step. However, since you can only link to output properties using the API testing UI, you must add an event handler to link the properties. - *Content.* The content comes from the response of the Web service. As this content is created dynamically during the test run, you must use an event handler to access the content.
Output Properties	None
Checkpoints	None

Events:

For this step, you need 2 events:

- A BeforeExecuteStepEvent event: This event links the *File Path* property of the current step to the *Folder Path* property from the CreateFile step. The code passes the value specified for the folder path to this test step so that the WriteToFile operation writes in the same folder and file that you created.

- An AfterExecuteStepEvent event: In this event, you access the response XML from the Web service, and use the code to extract the binary data from the Web service response. Then, you add the binary data as the content for the WriteToFile operation, in the folder and file specified in the BeforeExecuteStepEvent event.

Tasks

How to Open a Window for Writing Custom Code

Relevant for: API testing only

When creating or editing a test, you can use event handlers to test non-standard behavior of your application's API. For non-custom code activities, the default event handlers include events for checkpoints, before step execution, and after step execution.

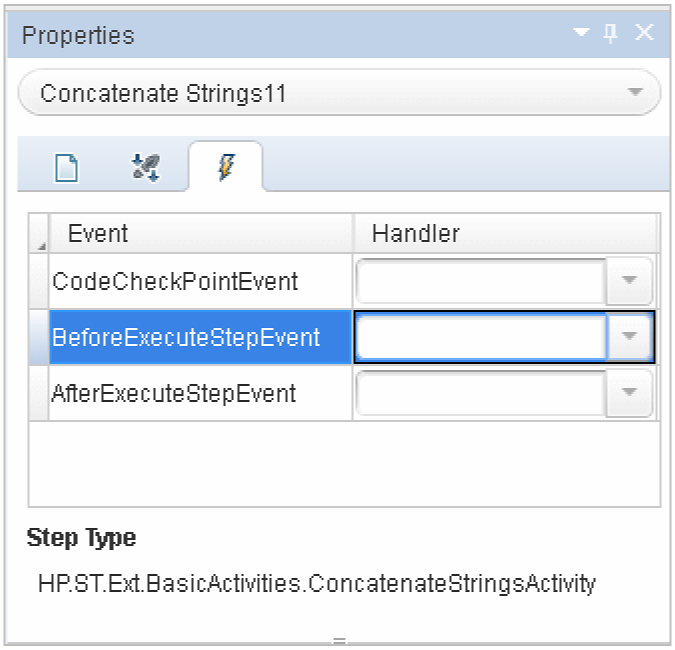
This task includes the following steps:

- "Open the Events tab" below
- "Select an event" on the next page
- "Edit the code" on the next page
- "Save the changes" on the next page

1. Open the Events tab

a. In the canvas, select an activity.

b. In the Properties pane, open the **Events** tab .



Concatenate Strings11

Event	Handler
CodeCheckPointEvent	
BeforeExecuteStepEvent	
AfterExecuteStepEvent	

Step Type

HP.ST.Ext.BasicActivities.ConcatenateStringsActivity

> **Note:** You can also open the Events tab by double-clicking a **Custom Code** activity in the canvas.

2. ## Select an event

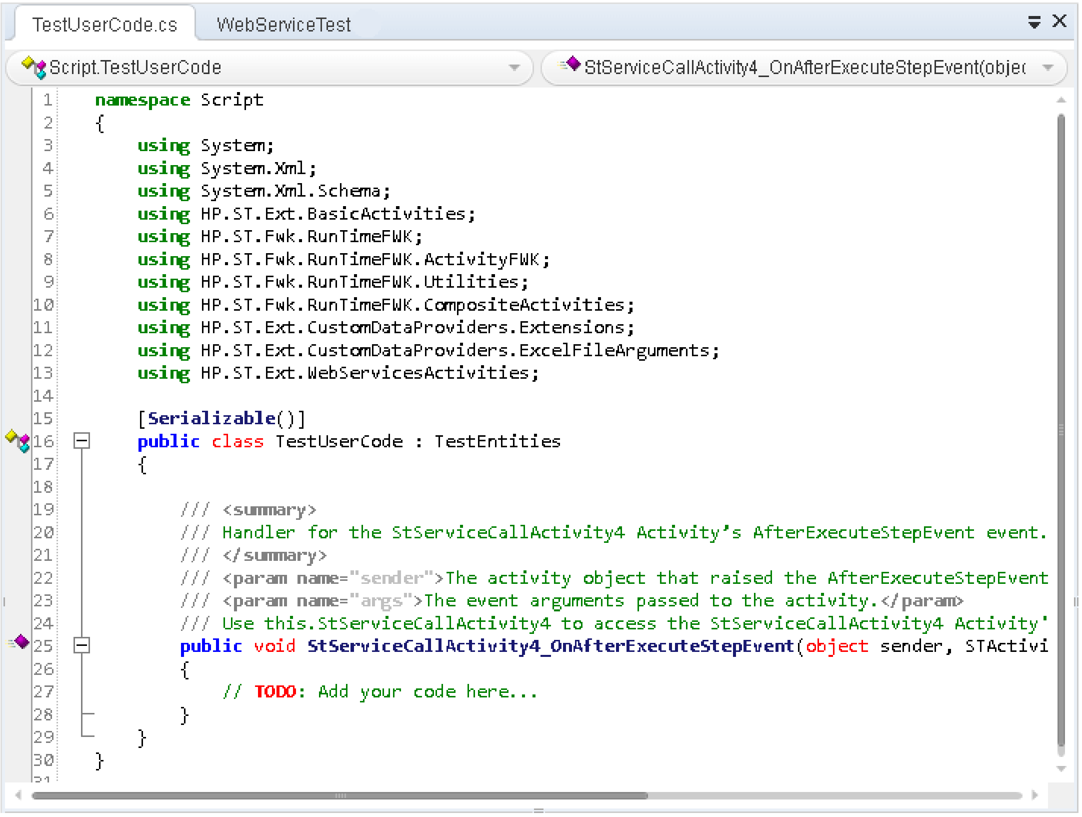
   In the Handler column, double-click the row for the event to which you want to provide code.

   The TestUserCode.cs file opens as a separate tab.

3. ## Edit the code

   In the TestUserCode.cs tab, locate the **TODO** section for your event handler and add your custom code.

> **Note:** Changes you make in the canvas are not reflected in the event handler code Intellisense/autocomplete options until you save the document.



```
namespace Script
{
    using System;
    using System.Xml;
    using System.Xml.Schema;
    using HP.ST.Ext.BasicActivities;
    using HP.ST.Fwk.RunTimeFWK;
    using HP.ST.Fwk.RunTimeFWK.ActivityFWK;
    using HP.ST.Fwk.RunTimeFWK.Utilities;
    using HP.ST.Fwk.RunTimeFWK.CompositeActivities;
    using HP.ST.Ext.CustomDataProviders.Extensions;
    using HP.ST.Ext.CustomDataProviders.ExcelFileArguments;
    using HP.ST.Ext.WebServicesActivities;

    [Serializable()]
    public class TestUserCode : TestEntities
    {

        /// <summary>
        /// Handler for the StServiceCallActivity4 Activity's AfterExecuteStepEvent event.
        /// </summary>
        /// <param name="sender">The activity object that raised the AfterExecuteStepEvent
        /// <param name="args">The event arguments passed to the activity.</param>
        /// Use this.StServiceCallActivity4 to access the StServiceCallActivity4 Activity'
        public void StServiceCallActivity4_OnAfterExecuteStepEvent(object sender, STActivi
        {
            // TODO: Add your code here...
        }
    }
}
```

4. ## Save the changes

   Click **File > Save All** to save the custom code and the test.

How to Manipulate Web Service Call/HTTP Request/SOAP Request Step Input/Output Properties

Relevant for: API testing only

Using code, you can access and set the properties of your HTTP/SOAP Request or Web Service steps.

This task includes the following activities:

- "Access and set property values for input properties" below
- "Add checkpoint property values" on the next page
- "Specify a SOAP Fault and SOAP Fault values" on page 1952
- "Assign a specific request file to a test step" on page 1953
- "Assign a specific request file to a Web service step in the OnSendRequest event" on page 1953
- "Set asynchronous Web service call properties" on page 1954
- "Add an input attachment to a Web service call" on page 1955
- "Access an attachment from a Web service call response" on page 1957
- "Add a HTTP Header for Web Service Calls" on page 1958
- "HTTP Headers for REST Service Calls" on page 1959

Access and set property values for input properties

1. In the canvas, select a Web service or SOAP Request step.
2. If you are using a SOAP Request step, load the XML containing the body of your SOAP request:

   a. In the Properties pane, open the **XML Body** tab .

   b. In the XML Body tab, click the **Load XML** button and navigate to your request file.

3. In the Properties pane, open the **Events** tab .
4. In the Events tab, create an event handler for the **AfterGenerateRequest** event. The TestUserCode.cs file opens.

5. In the **TODO** section of the TestUserCode.cs file, add the property value, using the following syntax:

   ```
   this.StServiceCallActivity<activity #>.InputEnvelope.SelectSingleNode(XPath to property).InnerText = "<value>";
   ```

   For details on the Input Envelope object, see "InputEnvelope Object" on page 1998. For details on the SelectSingleNode method, see "SelectSingleNode Method" on page 2018.

Example

The following example assigns the value of the DepartureCity input property for a flight booking web service from an Excel data source:

```
string newDepatureCityValue = GetDataSource("SampleAppData!Input").GetValue(this.Loop2.CurrentIterationNumber-1, "DepartureCity").ToString();
string departureCityXpath = "/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='DepartureCity'][1]";
this.StServiceCallActivity4.InputEnvelope.SelectSingleNode(departureCityXpath).InnerText = newDepatureCityValue;
```

Add checkpoint property values

You can use code to add checkpoint values in a Web service or SOAP request step. This can be very useful if the response on which the checkpoints is based is dynamically created.

Note: You cannot change the value of already existing checkpoints set in the Web service call.

1. Create an event handler for the CodeCheckpointEvent, as described in "Set the value of a checkpoint" on page 1970.

2. Following the line containing your code for enabling the checkpoint (args.Checkpoint.RunUICheckpoints = true), enter the value of your checkpoint, using the following syntax:

   ```
   args.Checkpoint.Assert.Equals(<actual value>,<expected value>);
   ```

   In the <actual value> and <expected value> parameters, you access the checkpoint properties through the step's output envelope. To access the output parameters, use the same syntax as described in "Access and set property values for input properties" on the previous page. However, you must change the InputEnvelope to OutputEnvelope to access the correct properties.

Example

In the following example, the checkpoint value for the DepartureCity value in a flight booking Web service is set:

```
string departureCityActualValueXpath = "/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlightsResponse'][1]/*[local-name(.)='GetFlightsResult'][1]/*[local-name(.)='Flight'][1]/*[local-name(.)='DepartureCity'][1]";
string ActualValue = this.StServiceCallActivity4.OutputEnvelope.SelectSingleNode(departureCityActualValueXpath).InnerText;

string departureCityExpectedValueXpath = "/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='DepartureCity'][1]";
string ExpectedValue = this.StServiceCallActivity4.InputEnvelope.SelectSingleNode(departureCityExpectedValueXpath).InnerText;

args.Checkpoint.Assert.Equals(ActualValue, ExpectedValue);
```

Specify a SOAP Fault and SOAP Fault values

Using code, you can set your Web service call or SOAP Request to expect a SOAP fault, as well as to specify the expected fault properties.

1. In the canvas, select a Web service or SOAP Request step.

2. In the Properties pane, open the **Events** tab ⚡.

3. In the Events tab, create an event handler for the AfterGenerateRequest event. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, specify the expected fault, using the following syntax:

   ```
   this.<activity>.FaultExpected = true;
   ```

5. In the Events tab, create an event handler for the CodeCheckpointEvent event.

6. In the **TODO** section of the TestUserCode.cs file, in the **CodeCheckpointEvent** section, specify the expected fault information, using the following syntax:

   ```
   string xpath = "<path to fault property>";
   string actualValue = this.StServiceCallActivity<activity #>.OutputEnvelope.SelectSingleNode(xpath).InnerText;
   string expectedValue = "soap:Server";
   ```

```
args.Checkpoint.Assert.Equals(actualValue,expectedValue);
```

> **Note:** For the specified string names in the syntax above, you can use your own names.

Example

```
string xpath = "/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='Fault'][1
]/*[local-name(.)='faultcode'][1]";
string actualValue = this.StServiceCallActivity4.OutputEnvelope.SelectSingleNode(xpath).InnerT
ext;
string expectedValue = "soap:Server";
args.Checkpoint.Assert.Equals(actualValue,expectedValue);
```

Assign a specific request file to a test step

1. In the canvas, select a Web service or SOAP Request step.
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler for the **AfterGenerateRequest** event. The TestUserCode.cs file opens.
4. In the **TODO** section of the TestUserCode.cs file, specify the expected fault, using the following syntax:

   ```
   this.<activity>.InputEnvelope.LoadXml(@"<path to response file>");
   ```

   > **Note:** You can also load the response from a previous step instead of from a file. In this case, you need to access the OutputEnvelope from a previous step in place of the @"<path to response file>" string. For details on accessing an output property from a step, see "Set the value of a checkpoint" on page 1970.

Assign a specific request file to a Web service step in the OnSendRequest event

1. In the canvas, select a Web service or SOAP Request step.
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler for the **OnSendRequest** event. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, specify the expected fault, using the following syntax:

   ```
   System.Xml.XmlDocument envelope = new XmlDocument();
   envelope.LoadXml(System.Text.Encoding.UTF8.GetString(args.Message));

   string xpath = "<fully qualified XPath to Property>";
   envelope.SelectSingleNode(xpath).InnerText = "<value to enter>";

   args.Message = System.Text.Encoding.UTF8.GetBytes(envelope.OuterXml);
   ```

Example

```
// Load request envelope into XML document
System.Xml.XmlDocument envelope = new XmlDocument();
envelope.LoadXml(System.Text.Encoding.UTF8.GetString(args.Message));

// Find and change the required node
string xpath = "/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='EchoAr
r'][1]/*[local-name(.)='arr'][1]/*[local-name(.)='int'][1]";
envelope.SelectSingleNode(xpath).InnerText = "10";

// Save changed envelope back
args.Message = System.Text.Encoding.UTF8.GetBytes(envelope.OuterXml);
```

Set asynchronous Web service call properties

1. In the canvas, select a Web service or SOAP Request step.

2. In the Properties pane, open the **Events** tab .

3. In the Events tab, create an event handler for **AfterGenerateRequest** event. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, specify the asynchronous call, using the following syntax:

   ```
   <activity name>.IsAsync = true;
   ```

5. Below the code for the IsAsync, specify the port on which to listen for the Web service or SOAP Request response, using the following syntax:

   ```
   this.<activity>.ListenOnPort = <port number>;
   ```

Example

```
StServiceCallActivity8.IsAsync = true;
this.StServiceCallActivity8.ListenOnPort = 8822;
```

Add an input attachment to a Web service call

You can use code to send an attachment with a Web service call. This is very useful when the attachment is generated dynamically, and you cannot add this attachment using the API testing interface during test design.

Note: If you are loading an attachment from outside your test, skip to step 5.

1. Add a custom code step to the canvas.
2. In the Properties pane, open the **Input/Output Properties** tab .
3. In the Input/Output Properties, click **Add** and select **Add Input Parameter**. In the Add Input Parameter dialog box, give the parameter a meaningful name.
4. Link the parameter to the desired attachment. For details, see "How to Assign Data to API Test/Component Steps" on page 1819.
5. In the canvas, select a Web service or SOAP Request step.
6. In the Properties pane, open the **Events** tab .
7. In the Events tab, create the **AfterGenerateRequest** event. The TestUserCode.cs file opens.
8. In the **TODO** section of the TestUserCode.cs file, specify the attachment to add, using the following syntax:

```
string attachmentsInfo =
    @"<InputAttachments>
        <Type> <attachment type> </Type>
        <Attachments>
            <Origin> <path to file> </Origin>
            <OriginType> File </OriginType>
            <ContentType> <type of content> </ContentType>
            <ContentID>Auto</ContentID>
        </Attachments>
    </InputAttachments>";

this.StServiceCallActivity<activity #>.InputAttachments = new XmlDocument();
this.StServiceCallActivity<activity #>.InputAttachments.LoadXml(attachmentsInfo);
```

You must define the attachment properties before the code that adds the attachment, as seen in the @<InputAttachments> of the code. These properties are then displayed for the test step in the Attachments tab in the Properties pane. For details on the necessary property values for attachments, see "Attachments Tab (Properties Pane - API Testing)" on page 403.

Note: You can define multiple attachments between the opening <InputAttachments> tag and the closing </InputAttachments> tag, using the syntax displayed above (between the <Attachments> and </Attachments> tags.

9. Optional - to override an attachment already defined in the Attachments tab in the Properties pane, you can use the following syntax:

```
string <string name> = "<fully qualified XPath to attachment defined in the Attachments tab>";
this.StServiceCallActivity<activity #>.InputAttachments.SelectSingleNode(<string name>).InnerText = @"<path to file>";
```

Note: This does not update the attachment's properties, but simply overwrites the attachment file.

Examples:

- The following example adds two text file attachments to your test:

  ```
  string attachmentsInfo =
      @"<InputAttachments>
          <Type>DIME</Type>
          <Attachments>
              <Origin>C:\somefile1.txt</Origin>
              <OriginType>File</OriginType>
              <ContentType>text/plain</ContentType>
              <ContentID>Auto</ContentID>
          </Attachments>
          <Attachments>
              <Origin>C:\somefile2.txt</Origin>
              <OriginType>File</OriginType>
              <ContentType>text/plain</ContentType>
              <ContentID>Auto</ContentID>
          </Attachments>
      </InputAttachments>";

  this.StServiceCallActivity5.InputAttachments = new XmlDocument();
  this.StServiceCallActivity5.InputAttachments.LoadXml(attachmentsInfo);
  ```

- The following example replaces an existing attachment with a text file:

  ```
  string firstAttachmentOriginXpath = "/*[local-name(.)='InputAttachments'][1]/*[local-name(.)='
  Attachments'][1]/*[local-name(.)='Origin'][1]";
  this.StServiceCallActivity5.InputAttachments.SelectSingleNode(firstAttachmentOriginXpath).
  InnerText = @"c:\somefile.txt";
  ```

Access an attachment from a Web service call response

By default, UFT saves attachments from a Web server response in the run results folder contained within the test's folder. However, you can also access these attachments using an event handler:

1. In the canvas, select the Web service or SOAP Request step for which you want to save the Web service call.

2. In the Properties pane, open the **Events** tab ⚡ .

3. In the Events tab, create an event handler for the **AfterExecuteStepEvent** event. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, access the attachment information, using the following syntax:

   ```
   string <string name> = System.IO.Path.Combine(this.StServiceCallActivity<activity #>.Cont
   ext.ReportDirectory,"Attachments");
   string[] <name> = System.IO.Directory.GetFiles(<string name>);
   ```

   This event handler returns an array which contains the full paths to the attachments returned with the Web service response.

Example

```
string responseAttachmentsFolder = System.IO.Path.Combine(this.StServiceCallActivity4.Conte
xt.ReportDirectory,"Attachments");
string[] responseAttachments = System.IO.Directory.GetFiles(responseAttachmentsFolder);
```

Add a HTTP Header for Web Service Calls

When you are editing your Web service call or SOAP Request steps, you can use an event to add an HTTP header. This is useful if the source for this information is created dynamically during a test run.

1. In the canvas, select a Web service or SOAP Request step.

2. In the Properties pane, open the **Events** tab ⚡ .

3. In the Events tab, create an event handler for the **BeforeApplyProtocolSettings** event. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, add the header element, using the following syntax:

   ```
   this.StServiceCallActivity4.HttpRequestHeaders.Add("<header key>", "< key value>");
   ```

Note: You can also set the HTTP headers values by expanding the **RequestHeader** node in the Properties pane's Input/Checkpoints tab. You then link to a data source from the **Name** and **Value** rows. For details, see "Select Link Source Dialog Box (API Testing)" on page 1840.

If you modify the headers using code in an event handler, it will override the values in the Properties pane during the test run.

HTTP Headers for REST Service Calls

To dynamically add an HTTP header to a REST service or to a REST service’s inner HTTP Request step (when working with API tests created in UFT 11.51 or earlier or Service Test 11.51 or earlier):

1. In the canvas, select a REST method step.
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler for the **BeforeExecuteStepEvent** event. The TestUserCode.cs file opens.
4. In the **TODO** section of the TestUserCode.cs file, allocate the desired length for the array:

   ```
   (args.Activity as HTTPActivity).RequestHeaders =
   new HP.ST.Shared.Utilities.Pair<string, string>[<# of headers>];
   ```

   ```
   (args.Activity as HTTPActivity).RequestHeaders = new HP.ST.Shared.Utilities.Pair<string, s
   tring>[2];
   ```

5. Below the allocation code, provide the each array element with the **<HeaderName>** and **<HeaderValue>** for each array element:

   ```
    (args.Activity as HTTPActivity).RequestHeaders[0] = new HP.ST.Shared.Utilities.Pair<strin
   g, string>("<header name>", "<header value>")
   ```

   Note: You will need to provide a separate line for each of the headers, as specified in your allocation statement.

Example

```
(args.Activity as HTTPActivity).RequestHeaders = new HP.ST.Shared.Utilities.Pair<string, strin
g>[2];
 (args.Activity as HTTPActivity).RequestHeaders[0] = new HP.ST.Shared.Utilities.Pair<string, stri
ng>"HeaderName1", "Value1");
 (args.Activity as HTTPActivity).RequestHeaders[1] = new HP.ST.Shared.Utilities.Pair<string, stri
ng>("HeaderName2", "Value2");
```

How to Stop an API Test Run from an Event

Relevant for: API testing only

Using custom code, you can stop a test run. This is useful if a condition in your test fails, and you do not want to continue the test run.

To stop the test run, do the following:

1. In the canvas, select the step on which you want to stop the test run (if necessary).
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler. T he TestUserCode.cs file opens. The event you select depends on the test property you want to ensure is correct and where that property occurs in the test flow.

   > **Tip:** To differentiate this event from other default events, enter a descriptive name like stopTestScript in the event name field.

4. In the **TODO** section of the TestUserCode.cs file, enter an if statement for the current activity.

   For details on if statements in C#, see http://msdn.microsoft.com/en-us/library/5011f09h(v=vs.90).aspx.

5. Below the if statement in the TestUserCode.cs file, enter the value you want to report using the following syntax:

   ```
   this.Context.ReplayApiClient.Stop();
   ```

If the condition entered in your event is not met, UFT immediately stops the test run, and does not display test results. If the condition is met, the test run continues.

> **Example**
>
> The following example stops a test run on a SOAP Response step. This code is set on the OnReceiveResponse event, given the name stopTestScript. It stops the test run if the SOAP response returns a fault:
>
> ```
> String node_xpath = "/*local-name(.)='Envelope'][1]/*local-name(.)='Body'][1]";
> if(this.StServiceCallActivity10.OutputEnvelope.SelectSingleNode(node_xpath).FirstChild.Name
> =="soapenv:Fault")
>
>     this.Context.ReplayApiClient.Stop();
> ```

How to Manipulate Data Programmatically

Relevant for: API testing only

Using code, you can retrieve or set test step property/parameter values, import or export property/parameter values, or data-drive the property/parameter values of your test steps.

This task includes the following steps:

- "Prerequisite" below
- "Retrieve a value from a data source" below
- "Set a property value from a data source" on the next page
- "Import a data source file to your test" on page 1963
- "Export the property values to a file" on page 1964
- "Data drive test step property/parameter values" on page 1964

Prerequisite

Add at least one data source to your test and save the test.

Retrieve a value from a data source

In order to get a value from a data source, you must use the GetDataSource and GetValue methods:

1. Select the step for which you want to retrieve a data source value.
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler. The TestUserCode.cs file opens.
4. In the **TODO** section of the TestUserCode.cs file, call the data source value you want to retrieve using the following syntax:

   ```
   GetDataSource("<data source name>".GetValue(<row index>, "<column name>");
   ```

   > **Note:** When entering the parameters for the GetValue function, your row index is based on 0.

5. (Optional) If you want to retrieve the value corresponding to the value in the current iteration, you replace the <row index> with the CurrentIterationNumber property, using the following syntax:

   ```
   GetDataSource(<data source name>.GetValue(<this.Loop<#>.CurrentIterationNumber, "<c
   olumn name>");
   ```

   > **Notes:**

- When running a data-driven test, UFT runs one iteration for each row in the data source. Therefore, the loop number you enter corresponds to the row of the data source, unless you specify a different starting row in the Data Source navigation policies.

- The CurrentIterationNumber is one-based, meaning that the number entered for the current loop must be 1 or higher.

Examples

- This example retrieves a value from the first row of an Excel data source, converts it to a string, and then assigns it as the property value for GetFlights test step Flight Number property:

  ```
  this.GetFlights4.FlightNumber =
  GetDataSource("GetFlights4_Input!MainDetails").GetValue(0, "Flight_Number").ToString(
  );
  ```

- This example also retrieves a value from an Excel data source, but from the current iteration's row in the Excel sheet. The event then assigns the retrieved value to the GetFlights test step's FlightNumber property.

  ```
  this.GetFlights4.FlightNumber =
  GetDataSource("GetFlights4_Input!MainDetails").
  GetValue(this.Loop4.CurrentIterationNumber, "Flight_Number").ToString();
  ```

Set a property value from a data source

You can use the SetValue method to insert a property value in a data source. This enables you to populate a data source with manually-entered values or values taken from another source.

1. Select the step for which you want to set a data source value.

2. In the Properties pane, open the **Events** tab ⚡ .

3. In the Events tab, create an event handler. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, call the data source value you want to enter using the following syntax:

   ```
   GetDataSource("<data source name>").SetValue(<row index>, "<column name>", "<value to
   enter>");
   ```

Example

The following example writes values to the Flight_Number and Tickets_Ordered columns of an Excel data source attached to a test:

```
GetDataSource("CreateFlightOrder4_Input!MainDetails").SetValue(0, "Flight_Number", "Y22");
GetDataSource("CreateFlightOrder4_Input!MainDetails").SetValue(0, "Tickets_Ordered", "2");
```

Import a data source file to your test

You can import data to your test using the Import and ImportFromExcelFile (if you are importing an Excel file) methods. This enables you to add data in runtime and populate your property values with this data.

1. Select the step to which you want to import data values
2. .In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler. The TestUserCode.cs file opens.
4. In the **TODO** section of the TestUserCode.cs file, call the data source value you want to enter using the following syntax:

   ```
   ExcelFileImportInputArgs <name> = new ExcelFileImportInputArgs(@"<path to data source>", "<data source name>", <boolean whether there is a header>);
   GetDataSource("<data source name>").Import(<name>);
   ```

   or

   ```
   GetDataSource("<data source name>").ImportFromExcelFile(@"<path to data source>", "<data source name>", <boolean whether there is a header>);
   ```

Example

The following example imports an Excel Data source:

```
ExcelFileImportInputArgs a = new ExcelFileImportInputArgs(@"C:\DemoExcel.xls", "MainDetails", true);
GetDataSource("CreateFlightOrder4_Input!MainDetails").Import(a);
```

```
GetDataSource("CreateFlightOrder4_Input!MainDetails").ImportFromExcelFile(@"C:\DemoExcel.xls", "MainDetails", true);
```

Export the property values to a file

You can also export the data from a test step to an external file using the Export and ExportToExcelFile methods. This enables you to export values to a file that other test steps can access to provide values for their properties/parameters in runtime.

1. Select the step for which you want to export its data values
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler. The TestUserCode.cs file opens.
4. In the **TODO** section of the TestUserCode.cs file, call the data source value you want to enter using the following syntax:

   ```
   ExcelFileExportInputArgs <name> = new ExcelFileExportInputArgs(@"<path to file>");
   GetDataSource("<data source name>").Export(<name>);
   ```

   or

   ```
   GetDataSource("<data source name>").ExporttoExcelFile(@"<path to file>");
   ```

Example

The following example takes the values from the input properties for a CreateFlightOrder step and writes these to an Excel file:

```
ExcelFileExportInputArgs a = new ExcelFileExportInputArgs(@"C:\ExportedExcel.xls");
GetDataSource("CreateFlightOrder4_Input!MainDetails").Export(a);

GetDataSource("CreateFlightOrder4_Input!MainDetails").ExportToExcelFile(@"C:\ExportedExcel.xls");
```

Data drive test step property/parameter values

You can also data drive test step property/parameter values using code. This is useful in custom scenarios where you cannot link to your data source with the user interface options or you need to link to a data source created in the test runtime.

1. Link the Test Flow/test loop with the data source. For details, see "Add a data source to the Test Flow or test loop" on page 1826.
2. Set the Data Navigation policy for the data source. For details, see "How to Set the Data Source Navigation Properties" on page 1825.
3. In the canvas, select the step to data drive.
4. In the Properties pane, open the **Events** tab ⚡ .

5. In the Events tab, create an event handler. The TestUserCode.cs file opens.

   > **Tip:** If you are populating values for the currently selected test step, use the BeforeExecuteEvent step. This ensures that the properties/parameters are mapped to the appropriate data source values before the step is run.

6. Connect your property value to the data source using the following syntax:

   - For test step properties are not based on an array:

     ```
     var <variable name> = GetDataSource("<data source>").GetValue(<row index>, "<column
     name>").ToString();
     <activity name>.<property name> = <variable name>;
     ```

   - For test step properties based on an array:

     ```
     var <variable name> = GetDataSource("<data source>").GetValue(<row index>, "<column
     name>").ToString();
     <activity name>.InputEnvelope.SelectSingleNode("<fully qualified xpath to property value
     >").InnerText = <variable name>;
     ```

   > **Notes:**
   >
   > - You can retrieve the XPath to a property name stored in an array by right-clicking the **Property** name in the **Input/Checkpoints** tab and selecting **Copy Fully Qualified XPath**.
   >
   > - If you want to set the value of an output value or checkpoint stored in an array, change the InputEnvelope to OutputEnvelope in the syntax displayed above.
   >
   > - If you are running multiple iterations using data-driving, you can use the this.Loop<#>.CurrentIterationValue property as the row index. However, since the CurrentIterationProperty is one-based, but the row-index is zero-based, add a -1 to the CurrentIterationProperty to ensure that your last iteration does not fail.

Example

The following example sets the **DepartureCity** and **ArrivalCity** values for a flight booking Web service from an Excel data source called "WebServiceData."

```
var GetFlights_Input_Departure_City = GetDataSource("WebServiceData!Input").GetValue(this.Loop2.CurrentIterationNumber-1, "DepartureCity").ToString();
StServiceCallActivity4.InputEnvelope.SelectSingleNode("/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='DepartureCity'][1]").InnerText = GetFlights_Input_Departure_City;
var GetFlights_Input_Arrival_City = GetDataSource("WebServiceData!Input").GetValue(this.Loop2.CurrentIterationNumber-1, "ArrivalCity").ToString();
StServiceCallActivity4.InputEnvelope.SelectSingleNode("/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='ArrivalCity'][1]").InnerText = GetFlights_Input_Arrival_City;
```

How to Access and Set the Value of Step Input, Output, or Checkpoint Properties

Relevant for: API testing only

You can set the values of input, output, or checkpoint properties through event handlers and custom code.

This task includes the following activities:

- "Access an step property" below
- "Access a step's parent activity" on page 1968
- "Set the value of a step's properties" on page 1968
- "Access the value of a step's property in runtime" on page 1969
- "Enable or ignore selected checkpoints - optional" on page 1970
- "Set the value of a checkpoint" on page 1970

Access an step property

1. From the Toolbox pane, drag the activity for which you want to access properties or add a custom code activity or event handler to an existing activity. Make sure that the custom code activity is after the activity for which you want to define properties.

2. In the canvas, select the step.

3. In the Properties pane, open the **Events** tab ⚡ .

4. In the Events tab, create an event handler. A TestUserCode.cs file opens in the document pane.

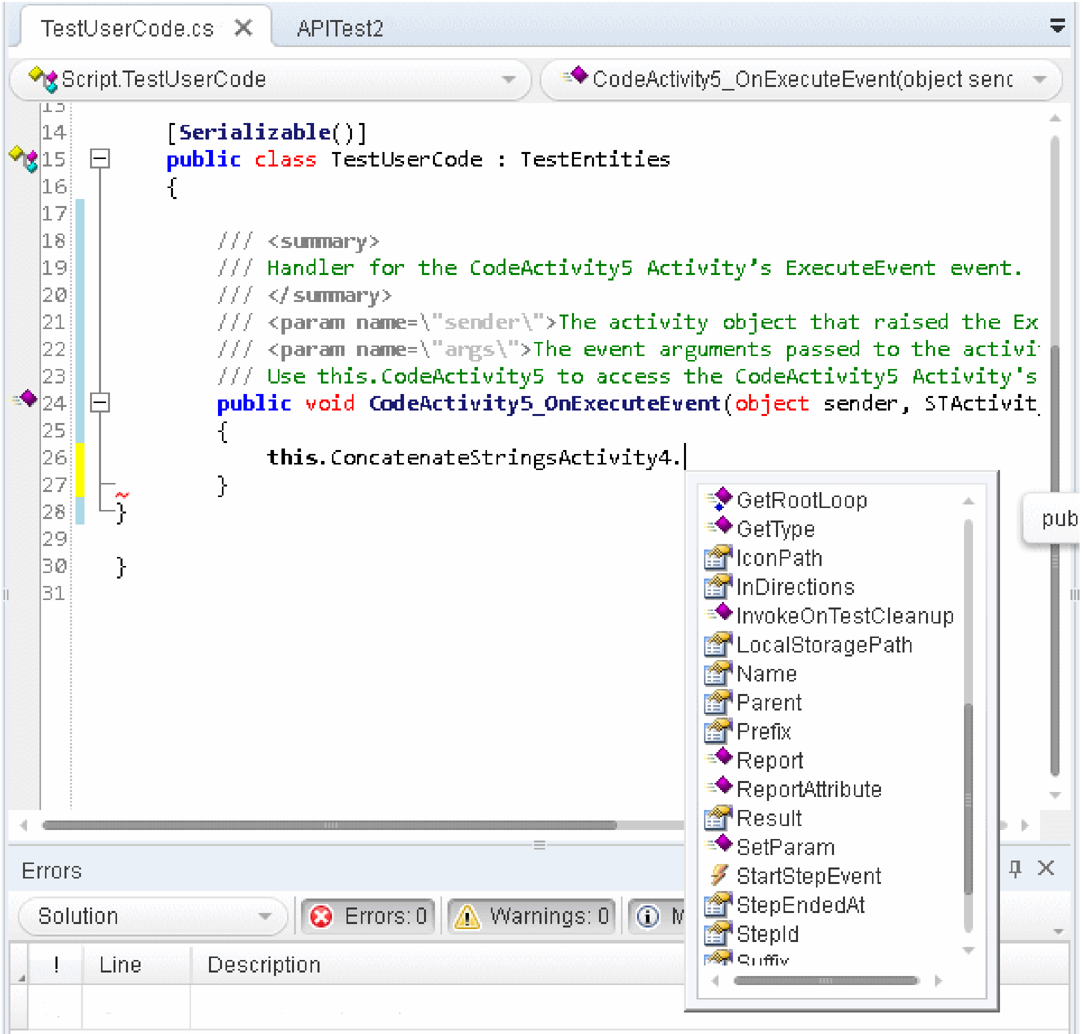
5. In the **TODO** section of the TestUserCode.cs file, use the this object to access the properties of the activity for which you want to set properties, using the following syntax:

```
this.<activity name>.Input.<property name>
```

> **Note:** For more details on the this object, see http://msdn.microsoft.com/en-us/library/dk1507sz.aspx.

6. Enter the step name for which you want to set properties followed by a . character.

7. Enter the property name of the activity to set properties, followed by a . character. In this example, you set the **Prefix** and **Suffix** properties for the ConcatenateStringsActivity step.



```
        [Serializable()]
        public class TestUserCode : TestEntities
        {

            /// <summary>
            /// Handler for the CodeActivity5 Activity’s ExecuteEvent event.
            /// </summary>
            /// <param name=\"sender\">The activity object that raised the Ex
            /// <param name=\"args\">The event arguments passed to the activi
            /// Use this.CodeActivity5 to access the CodeActivity5 Activity's
            public void CodeActivity5_OnExecuteEvent(object sender, STActivit
            {
                this.ConcatenateStringsActivity4.
            }
    }

    }
```

Access a step's parent activity

The **Parent** property of a step refers to the loop, condition, or parent activity that encloses a test step.

1. To access the activity's parent, use the this object as described in the "Access an step property" on page 1966 step,

2. Instead of entering the step's properties, enter the Parent property for the step.

3. If you want to get the parent loop for an activity, enter the GetParentLoop method, using the following syntax:

   ```
   this.<activity name>.GetParentLoop();
   ```

Example

This example retrieves the parent activity and converts it into a string.

```
string ParentName = this.ConcatenateStringsActivity5.Parent.Name
```

Set the value of a step's properties

1. In the canvas, select the step for which you want to set property values.

2. In the Properties pane, open the **Events** tab ⚡ .

3. In the Events tab, create an event handler. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, enter the parameter value for the parameters/properties, using this syntax:

   ```
   this.<activity name>.Input.<parameter name> = <parameter value>;

   or

   this.<activity name>.Output.<parameter name> = <parameter value>;
   ```

   You cannot set checkpoint values using the this.<activity name> object. You must use the args.Checkpoint object instead.

   > **Important:** Make sure that the value is the same type as the type entered when you created the parameter, i.e. a string parameter must have a string value.

When your test runs, UFT passes the value as specified in your custom code step to the other activity.

Example

This example sets the value of the Prefix and Suffix value for a Concatenate Strings activity:

```
CodeActivity6.Input.a = "Hello";
CodeActivity6.Input.b = "World";
this.ConcatenateStringsActivity4.Prefix = CodeActivity6.Input.a;
this.ConcatenateStringsActivity4.Suffix = CodeActivity6.Input.b;
```

Access the value of a step's property in runtime

You can also report the runtime value of a test step's property or parameter in the Output pane during a test run. This is useful if you want to watch the value of a particular operation or object during the test run, without having to stop and debug the test.

To do this, you can use a UserLogger object:

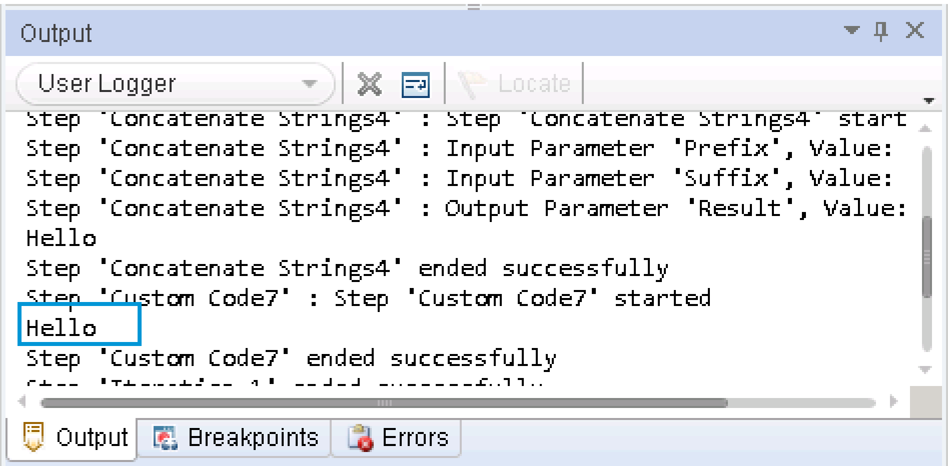
1. Add an event handler for the **OnAfterExecuteStep** event to the step for which you want to watch a property/parameter value.

2. Use the UserLogger object to report the value to the compilation log in the Output pane, using the following syntax:

   ```
   Context.UserLogger.Info(<activity name>.<property name>);
   ```

   Note: Using the Context object in an event handler or custom code enables you to get the contextual value for the property/parameter (i.e., the runtime value), instead of the entered value (what you enter in the Properties pane, for example).

When the compilation log is displayed in the Output pane, you will see a line with the property/parameter value displayed as part of the compilation log. Note that the Output pane does not list the property/parameter name with the value, so you must search within the step where you placed the code to find this value, as seen in the example below:



```
Step 'Concatenate Strings4' : Step 'Concatenate Strings4' start
Step 'Concatenate Strings4' : Input Parameter 'Prefix', Value:
Step 'Concatenate Strings4' : Input Parameter 'Suffix', Value:
Step 'Concatenate Strings4' : Output Parameter 'Result', Value:
Hello
Step 'Concatenate Strings4' ended successfully
Step 'Custom Code7' : Step 'Custom Code7' started
Hello
Step 'Custom Code7' ended successfully
Step 'Iteration 1' ended successfully
```

Example

This example retrieves the value of the Prefix property of a **ConcatenateStrings** activity:

```
Context.UserLogger.Info(ConcatenateStringsActivity4.Prefix);
```

Enable or ignore selected checkpoints - optional

You can instruct UFT to ignore the checkpoints for any test step. This is useful if you need to switch between enabling and ignoring checkpoints on different test runs.

You use the Checkpoint object to access the checkpoint's properties:

1. In the canvas, select the step for which you want to enable or ignore the checkpoints.

2. In the Properties pane, select the **Events** tab ⚡ .

3. In the Events tab, create an event handler for the **OnCodeCheckpointEvent** event. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, enable or ignore a checkpoint for an event using the following syntax:

   args.Checkpoint.RunUICheckpoints = true (to enable a checkpoint)

   or

   args.Checkpoint.RunUICheckpoints = false (to disable a checkpoint)

   For more details on the args object, see http://msdn.microsoft.com/en-us/library/aa884376(v=ax.50).aspx.

Set the value of a checkpoint

In addition to enabling or ignoring a checkpoint, you can also set the value (expected or not expected) of a checkpoint. This can be useful both to validate that your application works as expected but also does not allow unexpected behaviors.

1. Enable a checkpoint for the step, as described in "Enable or ignore selected checkpoints - optional" above.

2. Following the line containing your code for enabling the checkpoint (args.Checkpoint.RunUICheckpoints = true), enter the value of your checkpoint, using the following syntax:

   ```
   args.Checkpoint.Assert.Equals("<actual value>", "<expected value>");
   ```

For more details on the args object, see http://msdn.microsoft.com/en-us/library/aa884376(v=ax.50).aspx. For details on the supported methods for the args object in UFT, see "Assert Object" on page 1989.

Examples

- The following example sets the value of a checkpoint for a **ConcatenateStrings** step:

```
args.Checkpoint.Assert.Equals(this.ConcatenateStringsActivity4.Prefix+this.
ConcatenateStringsActivity4.Suffix, this.ConcatenateStringsActivity4.Result);
```

- This example checks if the text value (alphabetical order for a string) of the prefix is less than the suffix. To ensure that the checkpoint succeeds, enter a prefix value that is greater than the suffix, for example a prefix of **aa** and a suffix of **bb**.

```
 args.Checkpoint.Assert.Less("Concatenate test", this.
ConcatenateStringsActivity4.Prefix, this.ConcatenateStringsActivity4.Suffix,
     "The prefix is less than the suffix");
```

How to Report Test Run-Time Information

Relevant for: API testing only

Using custom events, you can report the run-time values or information of a given step, property, or parameter. You can choose either to report this to the Output pane or the Run Results Viewer. Viewing this information provides a simpler alternative to watching a object/property/parameter value while debugging.

To send run-time information, you can use the Report function or the UserLogger object.

This task includes the following activities:

- "Report a custom message to the run results" below
- "Report run-time values to the Output pane" on page 1973

Report a custom message to the run results

Using the Report function, you can send a custom message to the Run Results Viewer.

1. Select the step for which you want to report information.
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, enter the value you want to report using the following syntax:

   ```
   this.<activity name>.Report("<report information title>", "<reported data>");
   ```

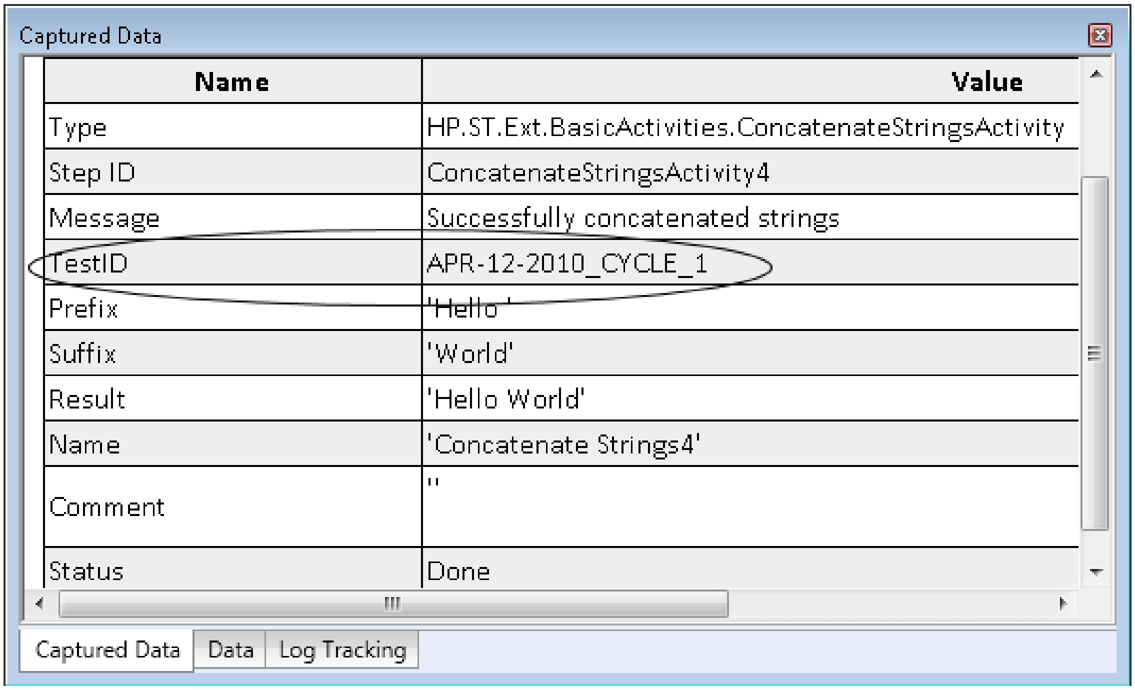
   or

   ```
   <activity name>.Report("<report information title>", "<reported data>");
   ```

   Tip: If you use the Context object after the <Activity Name> property. you report the run-time value of the selected object, property, or parameter.

The Report method displays a custom message in the Captured Data pane of the run results.In this example, a ConcatenateStrings test step implemented the following code in an event handler:

```
args.Activity.Report("TestID","APR-12-2010_CYCLE_1");
```



Captured Data

Name	Value
Type	HP.ST.Ext.BasicActivities.ConcatenateStringsActivity
Step ID	ConcatenateStringsActivity4
Message	Successfully concatenated strings
TestID	APR-12-2010_CYCLE_1
Prefix	'Hello '
Suffix	'World'
Result	'Hello World'
Name	'Concatenate Strings4'
Comment	''
Status	Done

Examples

The following example reports the value used for the Prefix property of a ConcatenateStrings activity:

```
this.ConcatenateStringsActivity4.Report("Run-time Prefix Value", this.ConcatenateStringsActivit
y4.Prefix)
```

Report run-time values to the Output pane

Using a UserLogger object, you can view the run-time value of a particular property, parameter, or object. This value is reported in the **UserLogger** build log displayed in the Output pane during test compilation.

UserLogger statements are useful in place of debugging. By viewing the run-time value of the selected property, parameter, or object, you can see the actual value without having to use the more complex debugging features.

1. Select the step for which you want to report information.

2. In the Properties pane, open the **Events** tab .

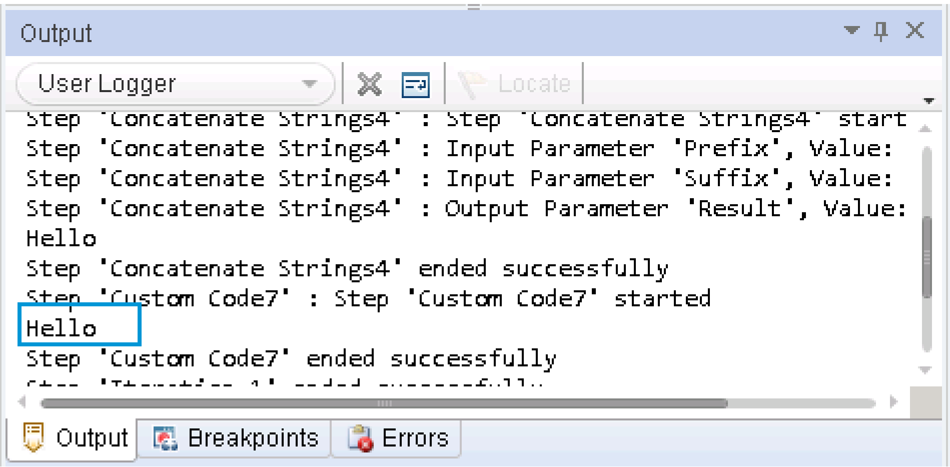
3. In the Events tab, create an event handler. The TestUserCode.cs file opens.

4. In the **TODO** section of the TestUserCode.cs file, enter the value you want to report using the following syntax:

   ```
   Context.UserLogger.<user logger info level>("<object/property/parameter>");
   ```

   > **IMPORTANT:** It is mandatory to use the Context object before the UserLogger object, as the Context object isolates the actual run-time context of the property, parameter, or object whose value you want to see.

You can then see the value you selected in the Output pane:



```
Step 'Concatenate Strings4' : Step 'Concatenate Strings4' start
Step 'Concatenate Strings4' : Input Parameter 'Prefix', Value:
Step 'Concatenate Strings4' : Input Parameter 'Suffix', Value:
Step 'Concatenate Strings4' : Output Parameter 'Result', Value:
Hello
Step 'Concatenate Strings4' ended successfully
Step 'Custom Code7' : Step 'Custom Code7' started
Hello
Step 'Custom Code7' ended successfully
Step 'Iteration 1' ended successfully
```

Examples

The following example reports the run-time value of the CustomerName property (whose value comes from a data source attached to the test) in a flight booking Web service. In this example the value to be reported was set with the DataSourceValue variable.

```
var DataSourceValue = GetDataSource("WebServiceData!Input").GetValue(0, "CustomerName");
Context.UserLogger.Info(DataSourceValue);
```

How to Retrieve and Set Test or User Variables

Relevant for: API testing only

You can use special code in test step events to retrieve or set the value of environment and user variables.

This task includes the following steps:

- "Prerequisite - create user variables." below
- "Optional - set the test profile" below
- "Retrieve a variable value" on the next page
- "Set a variable value" on the next page

Prerequisite - create user variables.

See "Define user variables" on page 142 for details on creating user variables.

Note: You may also want to create multiple test profiles to vary the value of the variables for different users or different test runs. For details on creating and editing user profiles, see "Define user variable profiles " on page 143

Optional - set the test profile

If you are using multiple test profiles, you must set the test profile you want to use before running test:

1. In the canvas, select either the **Start** or **End** steps.
2. In the Properties pane, open the **Test Variables** tab 🌐.
3. In the Test Variables tab, choose the profile name from the **Active Profile** drop-down list.

If you have entered default values for any test or user variables, the values for this profile are used in the test run.

Retrieve a variable value

You can use code to retrieve the value of a user variable during run-time. This is useful to show you the value of the variable without having to start a debugging session.

1. In the canvas, select the step during which you want to retrieve a variable value.
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler. The TestUserCode.cs file opens.
4. In the **TODO** section of the TestUserCode.cs file, call the data source value you want to retrieve using the following syntax:

   ```
   this.<activity name>.Context.TestProfile.GetVariableValue("<variable name>"); (if you
   want to use a specific value from a specific profile)
   ```

   or

   ```
   this.<activity name>.Context.EnvironmentProfile.GetVariableValue("<variable name>");
   ```

Example

```
this.ConcatenateStringsActivity4.Context.TestProfile.GetVariableValue("Prefix");
this.ConcatenateStringsActivity4.Report("Prefix Variable Value", this.ConcatenateStringsActivity
4.Context.TestProfile.GetVariableValue("Prefix"));
```

Set a variable value

1. In the canvas, select the step during which you want to retrieve a variable value.
2. In the Properties pane, open the **Events** tab ⚡ .
3. In the Events tab, create an event handler. The TestUserCode.cs file opens.
4. In the **TODO** section of the TestUserCode.cs file, call the data source value you want to retrieve using the following syntax:

   ```
   this.<activity name>.Context.TestProfile.SetVariableValue("<variable name>","<variable
   value>"); (if you want to enter a value for a specific test profile)
   ```

   or

   ```
   this.<activity name>.Context.EnvironmentProfile.SetVariableValue("<variable
   ```

```
name>","<variable value>");
```

Examples

- The following example retrieves the value of the **TestName** test variable:

  ```
  string testName = this.StServiceCallActivity4.Context.TestProfile.GetVariableValue("TestName");
  ```

- The following example sets the value of the environment variable beforeExecuteStepevent used to verify that the BeforeExecuteStepEvent event ran in the test step for the activity activity.

  ```
  activity.Context.EnvironmentProfile.SetVariableValue("beforeExecuteStepEvent","true");
  ```

How to Encrypt and Decrypt Passwords

Relevant for: API testing only

Password fields expect an encrypted string - if you provide an unencrypted string, the authentication will fail.

The **EncryptionMngr** method lets you encrypt and decrypt strings within your events.

This task includes the following steps:

- "Encrypt the password" below
- "Decrypt the password - optional" below

1. Encrypt the password

Use the activity's context's **EncryptionMngr** method, and select Encrypt from the autocompletion drop-down. The following example encrypts a password.

```
string plainText = "myPassword";
string encryptedText = this.StServiceCallActivity4.Context.EncryptionMngr.Encrypt(plainText);
```

2. Decrypt the password - optional

Use the **Decrypt** method from the autocompletion drop-down.

The following example decrypts a password and validates it against the original string.

```
string decryptedText = this.StServiceCallActivity4.Context.EncryptionMngr.Decrypt(encryptedText);
bool equalText = decryptedText.Equals(plainText);
```

Reference

API Event Coding - Event Structure Overview

Relevant for: API testing only

When adding event handlers for your API tests, you can choose from a number of different event handlers. Each of these event handlers runs at a different place in the test step's execution, and likewise can access different properties of the current activity.

For most of the API activities included in UFT, you can choose from the standard event handlers: BeforeExecuteStepEvent, AfterExecuteStepEvent, or CodeCheckpointEvent. The function of these is the same regardless of which test step they are used with or the unique properties for each test step.

For Web service and SOAP Request activities, you can access additional event handlers that enable you to add specific functionalities to the Web service call or SOAP request.

The following sections detail the possible events you can use when adding event handlers:

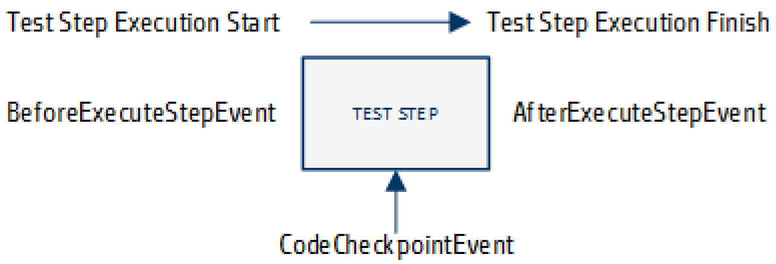
API Event Coding - Standard Event Structure

Relevant for: API testing only

For the majority of the activities supported for an API test in UFT, you can only use the standard event handlers:

- BeforeExecuteStepEvent
- AfterExecuteStepEvent
- CodeCheckpointEvent

The following diagram shows how each event works within the individual test step run:



Because of this design, each activity has a different purpose and has access to different properties of the individual test step:

- "BeforeExecuteStepEvent" below
- "AfterExecuteStepEvent" on the next page
- "CodeCheckpointEvent" on page 1981

BeforeExecuteStepEvent

Purpose: Set conditions and properties of the step required to make the step run or to handle output from a previous step required in the current step

Accessible Properties:

- Input properties/parameters from the current activity
- User/test variables from the current test
- Output properties from a previous test step or a parent activity

Example

```
/// <summary>
    /// Handler for the ConcatenateStringsActivity4 Activity’s BeforeExecuteStepEvent event.
    /// </summary>
    /// <param name=\"sender\">The activity object that raised the BeforeExecuteStepEvent event
.</param>
    /// <param name=\"args\">The event arguments passed to the activity.</param>
    /// Use this.ConcatenateStringsActivity4 to access the ConcatenateStringsActivity4 Activity's c
ontext, including input and output properties.
    public void ConcatenateStringsActivity4_OnBeforeExecuteStepEvent(object sender, STActivit
yBaseEventArgs args)
    {
        ExcelFileImportInputArgs a = new ExcelFileImportInputArgs(@"C:\Users\user\Documents\Un
ified Functional Testing\API Test Resources\ConcatenateStrings.xlsx", "Sheet1", true);
            GetDataSource("ConcatenateStrings!Sheet1").ImportFromExcelFile(@"C:\Users\user\Docu
ments\Unified Functional Testing\API Test Resources\ConcatenateStrings.xlsx", "Sheet1", true);
        ConcatenateStringsActivity4.Prefix = GetDataSource("ConcatenateStrings!Sheet1").GetValu
e(0, "Prefix").ToString();
        ConcatenateStringsActivity4.Suffix = GetDataSource("ConcatenateStrings!Sheet1").GetValu
e(0, "Suffix").ToString();
        this.Context.UserLogger.Info (ConcatenateStringsActivity4.Prefix);
        this.Context.UserLogger.Info (ConcatenateStringsActivity4.Suffix);

    }
```

AfterExecuteStepEvent

Purpose: Set output properties for the current step or handle the output of the step (to export, save, etc.)

Accessible Properties:

- Output properties/parameters from the current activity
- User/test variables from the current test
- Any output from the current test step

Example:

```
/// <summary>
    /// Handler for the FileWriteActivity7 Activity’s AfterExecuteStepEvent event.
    /// </summary>
    /// <param name=\"sender\">The activity object that raised the AfterExecuteStepEvent event.<
/param>
    /// <param name=\"args\">The event arguments passed to the activity.</param>
    /// Use this.FileWriteActivity7 to access the FileWriteActivity7 Activity's context, including input
and output properties.
    public void FileWriteActivity7_OnAfterExecuteStepEvent(object sender, STActivityBaseEvent
Args args)
    {
            ///Event code is here
            String WriteToFileContent = this.StServiceCallActivity5.OutputEnvelope.SelectNodes("/*[loca
l-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='CreateFlightOrder'][1]").ToSt
ring();
            this.FileWriteActivity7.Content = WriteToFileContent;
```

CodeCheckpointEvent

Purpose: To set checkpoint properties and values for the current step

Accessible Properties:

- Input and output values for the current step
- User test/variables from the current test.
- All input and output from the current test step

Example:

```
 /// <summary>
     /// Handler for the CodeActivity15 Activity?s CodeCheckPointEvent (General execute event f
or executing code checkpoints) event.
     /// </summary>
     /// <param name="sender">The activity object that raised the CodeCheckPointEvent event.<
/param>
     /// <param name="args">The event arguments passed to the activity.</param>
     /// Use args to access the CodeActivity15 Activity's context, including any input and output ar
guments.
     /// <example></example>
     public void CodeActivity15_OnCodeCheckPointEvent(object sender, CheckpointEventArgs a
rgs)
     {
        ///Event code starts here
               string attachPath = CodeActivity15.Input.attachmentPath;
        string attachmentContent = File.ReadAllText(attachPath);
        string currDate = DateTime.Now.ToString();

        currDate = currDate.Substring( 0,currDate.IndexOf(":")  ) ;
        //args.Checkpoint.Assert.Equals( attachmentContent.Contains(" Current Time = "+currDat
eTime+"   file was attached$") ,true);
        //bool status= attachmentContent.Contains(" Current Time = "+currDate);
        bool status = attachmentContent.Contains( CodeActivity16.Output.currentDateTime );
        bool status1 = attachmentContent.Contains("file was attached$");
        if (status && status1)
        {
         args.Checkpoint.Report( attachmentContent," Current Time = "+currDate+"   file was atta
ched$","contain",  StatusEnum.Succeed );
```

```
       }
       else
         args.Checkpoint.Report( attachmentContent," Current Time = "+currDate+"   file was attac
hed$","contain", StatusEnum.Fail );
    }
```

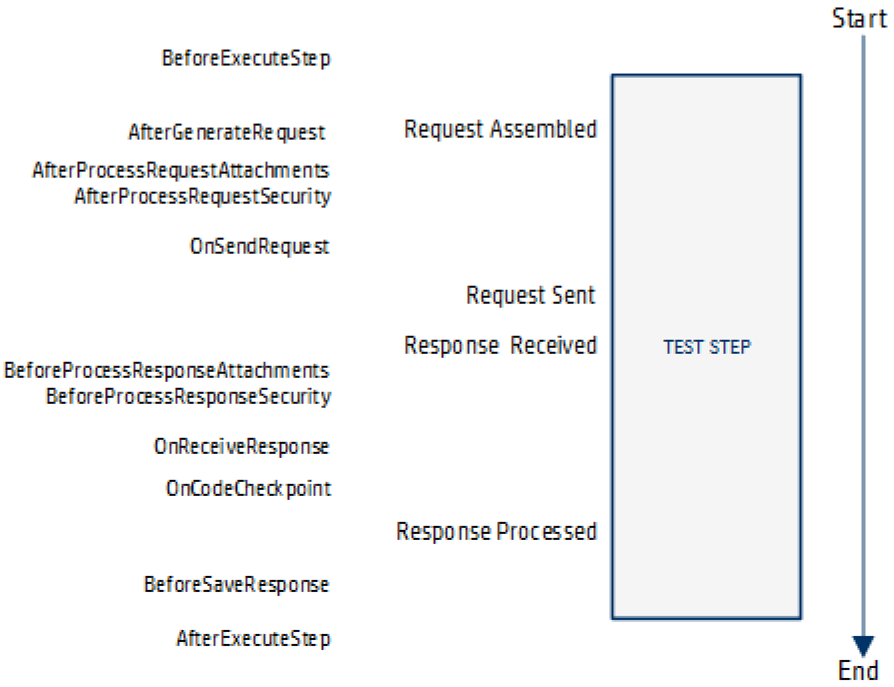
API Event Coding - Web Service Event Structure

Relevant for: API testing only

When working with a Web service call step or a SOAP Request step, there are a number of additional events available that correspond with the unique run structure of a Web service call:

- BeforeExecuteStepEvent
- AfterExecuteStepEvent
- CodeCheckpointEvent
- AfterGenerateRequest
- AfterProcessRequestSecurity
- AfterProcessRequestAttachments
- OnSendRequest
- OnReceiveResponse
- BeforeProcessResponseAttachments
- BeforeProcessResponseSecurity
- BeforeSaveResponse
- BeforeApplyProtocolSettings

The following diagram shows how each event works within the individual test step run:



However, due to the flow and timing of the various events, you should only create event handlers for specific events:

- "BeforeExecuteStepEvent" below
- "AfterExecuteStepEvent" on the next page
- "CodeCheckpointEvent" on the next page
- "AfterGenerateRequest" on the next page
- "AfterProcessRequestSecurity (WCF services only)" on page 1986
- "OnReceiveResponse" on page 1986
- "BeforeProcessResponseSecurity (WCF Security Scenarios only)" on page 1986
- "BeforeSaveResponse" on page 1986

BeforeExecuteStepEvent

Purpose: Set conditions and properties of the step required to make the step run or to handle output from a previous step required in the current step

Accessible Properties:

- Input properties/parameters from the current activity
- User/test variables from the current test
- Output properties from a previous test step or a parent activity

AfterExecuteStepEvent

Purpose: Set conditions and properties of the step required to make the step run or to handle output from a previous step required in the current step

Accessible Properties:

- Input properties/parameters from the current activity
- User/test variables from the current test
- Output properties from a previous test step or a parent activity
- Response data from the current step
- Response attachments from the current step

CodeCheckpointEvent

Purpose: Set conditions and properties of the step required to make the step run or to handle output from a previous step required in the current step

Accessible Properties:

- Input properties/parameters from the current activity
- User/test variables from the current test
- Output properties from a previous test step or a parent activity
- SOAP Fault properties

AfterGenerateRequest

Purpose: Set conditions and properties of the step required to make the step run or to handle output from a previous step required in the current step

Accessible Properties:

- Input properties from the current step
- The input envelope from the current step
- The input attachments from the current step
- Asynchronous properties from the current step

AfterProcessRequestSecurity (WCF services only)

Purpose: Update the request envelope information for Web services using a WCF security scenario with WSE defined. For details on the WCF security scenarios, see "Security Scenarios Overview" on page 1878.

Use the args.Message property to access the response envelope

Accessible Properties:

- Input envelope information for the current test.

OnReceiveResponse

Purpose: Access the output envelope for the current test for Web services using a Web Service security scenario with WSE defined. For details on the WCF security scenarios, see "Security Scenarios Overview" on page 1878.

Use the arg.Message property to access the response envelope

Accessible Properties:

- The response envelope information for the current step. When this runs, the Web service call step returns a byte array containing the response envelope. You must add event handler code also to use the byte array data.

  Use the arg.Message property to access the response envelope

BeforeProcessResponseSecurity (WCF Security Scenarios only)

Purpose: Access the output envelope for the current step for Web services using a WCF security scenario with WSE defined. For details on the WCF security scenarios, see "Security Scenarios Overview" on page 1878.

Use the arg.Message property to access the response.

Accessible Properties:

- The response envelope information for the current step.

BeforeSaveResponse

Purpose: Access the current step's response.

Accessible Properties:

- The response for the current step. Use the arg.Message property to access the response.

API Test Event Coding Common Objects

Relevant for: API testing only

When writing custom code for events in your API test step events, you can use the following objects:

Activity Object

Relevant for: API testing only

Description

Accesses the current activity's properties and arguments.

Syntax

args.**Activity**.<supported object/method>

Supported Methods

- Comment object
- Context object
- EnableReporting object
- GetParentLoop method
- GetRootLoop method
- Name object
- Parent object
- Report method
- StepEndedAt object
- StepId object (read-only)

Assert Object

Relevant for: API testing only

Description

Qualifies the preceding object.

> **Note:** This object should be used only in the CodeCheckpointEvent event.

Syntax

<object>.**Assert**.<supported operator method("<expected value>");

Supported Methods

- Equals
- Greater
- GreaterorEqual
- Less
- LessorEqual
- NotEquals

Example

```
args.Checkpoint.Assert.Equals(this.ConcatenateStringsActivity4.Prefix+this.ConcatenateStringsActivity4.Suffix,this.ConcatenateStringsActivity4.Result);
```

Checkpoint Object

Relevant for: API testing only

Description

Accesses the current activity's checkpoint properties.

> **Note:** This object is only accessible when using the OnCodeCheckpointEvent event.

Syntax

args.**Checkpoint**.<supported object/method>

> **Note:** You must use the args object before the Checkpoint object to access the selected activity's checkpoint properties.

Supported Objects and Methods

- Assert object
- RunUIcheckpoints object
- Report method

Examples

```
args.Checkpoint.RunUICheckpoints = true;
```

```
if (args.Checkpoint.Assert.Equals(this.ConcatenateStringsActivity4.Result))
   this.ConcatenateStringsActivity4.Report(ConcatenateStringsActivity4.Result, "Passed");
else
   this.ConcatenateStringsActivity4.Report(ConcatenateStringsActivity4.Result, "Failed");
```

Context Object

Relevant for: API testing only

Description

Accesses the run-time context of the preceding object.

> **Note:** You should add another object following this object to specify the specific context of the selected activity you want to access (i.e., follow this object with a Checkpoint object to isolate checkpoint properties for the activity.

Syntax

this.<activity name>.**Context**.<object>

Supported Methods

None

Example

```
Context.UserLogger.Info(ConcatenateStringsActivity4.Prefix);
```

CurrentIterationNumber Object

Relevant for: API testing only

Description

Gets the current iteration number.

Syntax

this.<parent activity>.**CurrentIterationNumber**

Supported Methods

None

Example

```
var GetFlights_Input_Arrival_City = GetDataSource("WebServiceData!Input").GetValue(this.Loo
p2.CurrentIterationNumber-1, "ArrivalCity").ToString();
StServiceCallActivity4.InputEnvelope.SelectSingleNode("/*[local-name(.)='Envelope'][1]/*[local-n
ame(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='ArrivalCity'][1]").InnerText = G
etFlights_Input_Arrival_City;
```

EncryptionMngr Object

Relevant for: API testing only

Description

Accesses UFT API testing's encryption mechanism to enable you to encrypt or decrypt strings (such as passwords).

Syntax

this.<activity>.Context.**EncryptionMngr**.<supported method>

Supported Methods

- Decrypt
- Encrypt

Example

```
string plainText = "myPassword";
string encryptedText = this.StServiceCallActivity4.Context.EncryptionMngr.Encrypt(plainText);
```

EnvironmentProfile Object

Relevant for: API testing only

Description

Accesses the environmental variables for a test.

> **Note:** If you are using test profiles for a test, you should use the **TestProfile** object instead.

Syntax

this.<activity>.Context.**EnvironmentProfile**.<supported method>

Supported Methods

- GetType
- GetVariablesNames
- GetVariableValue
- SetVariableValue
- ToString

Example

```
this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariablesNames().ToString();
```

InputAttachment Object

Relevant for: API testing only

Description

Accesses the properties of the test step's input attachments.

Syntax

this.<activity>.Context.**InputAttachments**.<supported object or method>

Supported Methods

- Attributes object
- BaseURL object
- ChildNodes object
- CreateAttribute method
- CreateCDataSection method
- CreateComment method
- CreateElement method
- CreateEntityReference method
- CreateNode method
- CreateWhitespace method
- CreateXMLDeclaration method
- DocumentElement object
- DocumentType object
- FirstChild object
- GetElementsbyId method
- GetElementsbyTagName method
- GetEnumerator method
- GetNamespaceOfPrefix method
- HasChildNodes object
- ImportNode method
- InnerText object
- InnerXML object
- InsertAfter method
- InsertBefore method

- IsReadOnly object
- LastChild object
- Load method
- LoadXML method
- LocalName object
- Name object
- NamespaceURL object
- NameTable object
- NextSibling object
- NodeType object
- OwnerDocument object
- ParentNode object
- Prefix object
- PrependChild method
- PreserveWhitespace object
- PreviousSibling object
- ReadNode method
- RemoveAll method
- RemoveChild method
- ReplaceChild method
- SchemaInfo object
- Schemas object
- SelectNodes method
- SelectSingleNode method
- Supports method
- Validate method

- Value object
- WriteContentTo method
- WriteTo method

Example

```
this.StServiceCallActivity8.InputAttachments.Load(@"<my file path>");
```

InputEnvelope Object

Relevant for: API testing only

Description

Accesses the input property envelope for Web Service, HTTP Request, and SOAP Request activities.

Syntax

this.<activity>.**InputEnvelope**.<supported object/method>

Supported Objects and Methods

This object is a standard object of the XML Document class. All supported objects and methods for the XML Document class can be used with this object.

Example

```
var GetFlights_Input_Arrival_City = GetDataSource("WebServiceData!Input").GetValue(this.Loop2.CurrentIterationNumber-1, "ArrivalCity").ToString();
StServiceCallActivity4.InputEnvelope.SelectSingleNode("/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='ArrivalCity'][1]").InnerText = GetFlights_Input_Arrival_City;
```

OutputAttachment Object

Relevant for: API testing only

Description

Accesses the properties of the test steps output attachments.

Syntax

this.<activity>.Context.**OutputAttachments.**<supported object or method>

Supported Methods

- Attributes object
- BaseURL object
- ChildNodes object
- CreateAttribute method
- CreateCDataSection method
- CreateComment method
- CreateElement method
- CreateEntityReference method
- CreateNode method
- CreateWhitespace method
- CreateXMLDeclaration method
- DocumentElement object
- DocumentType object
- FirstChild object
- GetElementsbyId method
- GetElementsbyTagName method
- GetEnumerator method
- GetNamespaceOfPrefix method
- HasChildNodes object
- ImportNode method
- InnerText object
- InnerXML object

- InsertAfter method
- InsertBefore method
- IsReadOnly object
- LastChild object
- Load method
- LoadXML method
- LocalName object
- Name object
- NamespaceURL object
- NameTable object
- NextSibling object
- NodeType object
- OwnerDocument object
- ParentNode object
- Prefix object
- PrependChild method
- PreserveWhitespace object
- PreviousSibling object
- ReadNode method
- RemoveAll method
- RemoveChild method
- ReplaceChild method
- SchemaInfo object
- Schemas object
- SelectNodes method
- SelectSingleNode method

- Supports method
- Validate method
- Value object
- WriteContentTo method
- WriteTo method

Example

```
this.StServiceCallActivity8.OutputAttachments.Load(@"<my file path>");
```

OutputEnvelope Object

Relevant for: API testing only

Description

Accesses the output envelope information for Web Service, HTTP Request, and SOAP Request activities.

Syntax

this.<activity>.**OutputEnvelope.**<supported object/method>

Supported Methods

This object is a standard object of the XML Document class. All supported objects and methods for the XML Document class can be used with this object.

Example

```
var GetFlights_Input_Arrival_City = GetDataSource("WebServiceData!Input").GetValue(this.Loop2.CurrentIterationNumber-1, "ArrivalCity").ToString();
StServiceCallActivity4.OutputEnvelope.SelectSingleNode("/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='ArrivalCity'][1]").InnerText = GetFlights_Input_Arrival_City;
```

Parent Object

Relevant for: API testing only

Description

Accesses the parent activity for a selected activity.

Syntax

this.<activity>.**Parent**.<supported method>

this.<activity>.Context.**Parent**.<supported method>

> **Note:** Using the Context object here enables you to access the run-time context of the selected activity and parent activity.

Supported Methods

- Activities object
- Comment object
- Context object
- EnableReporting object
- GetParentLoop
- GetRootLoop
- Name object
- Report method
- StepEndedAt object
- StepId object

Example

```
string ParentName = this.ConcatenateStringsActivity5.Parent.Name
```

TestProfile Object

Relevant for: API testing only

Description

Accesses the values of the environment and user variables for the currently active test profile as specified in the Properties pane.

Syntax

this.<activity>.Context.**TestProfile**.<supported method>

Supported Methods

- GetType
- GetVariableNames
- GetVariableValue
- SetVariableValue

Example

```
this.ConcatenateStringsActivity4.Context.TestProfile.GetVariableValue("Prefix");
```

UserLogger Object

Relevant for: API testing only

Description

Reports the specified conditions of the preceding object to the UserLogger build log in the Output pane.

Syntax

this.<activity>.Context.**UserLogger**.<user logger detail supported method>

Supported Methods

- Info
- InfoFormat
- Debug
- DebugFormat
- Error
- ErrorFormat
- Fatal
- FatalFormat
- Warn
- WarnFormat

For details on the supported methods, see http://www.codeproject.com/Articles/140911/log4net-Tutorial.

Example

```
var Variablenames = this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariablesNames();
Context.UserLogger.Info(Variablenames);
```

API Test Event Coding Common Methods

Relevant for: API testing only

When writing custom code for your API test activities, there are a number of methods that you can use:

Note: The **Query** function implemented in versions 11.20 and earlier is not supported in version 11.50 and later. To modify runtime data through an event handler, replace all occurrences of the **Query** function with **GetDataSource**, using the arguments described in this section..

Export Method

Relevant for: API testing only

Description

Exports the specified Excel data source from your test to an external file.

Class

DataSource

Syntax

```
ExcelFileExportInputArgs <name> = new ExcelFileImportInputArgs(@"<path to data source>");
GetDataSource("<data source name>").Export (name);
```

> **Note:** You must cast the data source in the first line of the syntax in order to use the Export method. If you do not want to cast the data source, use the **ExportToExcelFile** method.

Parameters

Parameter	Description
Name	Name assigned to the data source.
Path to data source	The Windows path to the file for the data source.

Return Type

An Excel file with the specified data in the specified directory.

Example

```
ExcelFileExportInputArgs a = new ExcelFileExportInputArgs(@"C:\Users\brojerem\Documents\Unified Functional Testing\API Test Resources\File.xls");
   GetDataSource("ConcatenateStrings!Sheet1").Export(a);
```

ExportToExcelFile Method

Relevant for: API testing only

Description

Exports the specified Excel data source from your test to an external file.

Class

DataSource

Syntax

GetDataSource("<data source>").**ExportToExcelFile**(@"<path to file>");

Parameters

Parameter	**Description**
Path to data source	The Windows path to the file for the data source.

Return Type

An Excel file in the specified directory.

Example

```
GetDataSource("ConcatenateStrings!Sheet1").ExportToExcelFile(@"C:\Users\brojerem\Documents\Unified Functional Testing\API Test Resources\File.xls");
```

GetDataSource Method

Relevant for: API testing only

Description

Accesses the specified data source.

> **Note:** This method is typically used when setting a value for a property, parameter, or variable. It is also used in conjunction with other methods which enable you to use the data from the data source, such as **GetValue** or **SetValue**.

Class

Test Entities

Syntax

property value = **GetDataSource**("<Data source name>").

Parameters

Parameter	Description
Data source name	The name of the data source, exactly as it appears in the Data pane. **Note:** You can right-click the data source in the Data pane and select Copy to ensure that you use the correct name in your code.

Return Type

No explicit return object - method only accesses the data source.

Example

```
var DataSourceValue = GetDataSource("WebServiceData!Input").GetValue(0, "CustomerName");
Context.UserLogger.Info(DataSourceValue);
```

GetValue Method

Relevant for: API testing only

Description

Retrieves a specified value from the selected data source.

Note: This method is a method of the data source. In order to use this method to get a data source value, you must retrieve a data source using the **GetDataSource** method

Class

DataSource

Syntax

GetDataSource(<data source name>).**GetValue**(<row index>, "<column name>");

Parameters

Parameter	Description
Row index	**Required.** The row from which to take the value. **Note:** The row index is zero-based, meaning if you want to pull from the first row of the table, you must set the row index value to 0.
Column name	**Required.** The name of the column from which to take the value.

Return Type

Data value (format depends based on the data source type)

Example

```
var TableDataSourceValue = GetDataSource("WebService Customer Table").GetValue(0, "CustomerName");
```

```
var GetFlights_Input_Departure_City = GetDataSource("WebServiceData!Input").GetValue
```

```
(this.Loop2.CurrentIterationNumber-1, "DepartureCity").ToString();
StServiceCallActivity4.InputEnvelope.SelectSingleNode("/*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='DepartureCity'][1]").InnerText
= GetFlights_Input_Departure_City;
```

GetVariableNames Method

Relevant for: API testing only

Description

Retrieves the names of all variable values in the current test.

Class

TestProfile

Syntax

this.<activity>.Context.TestProfile.**GetVariableNames**();

this.<activity>.Context.EnvironmentProfile.**GetVariableNames**();

Parameters

None

Return Type

A list of all environment/test variables in the current test or test profile.

Example

```
this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariablesNames().ToString();
var Variablenames = this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariables
Names();
Context.UserLogger.Info(Variablenames);
```

GetVariableValue Method

Relevant for: API testing only

Description

Retrieves the value of a selected test or environment variable.

Class

TestProfile

Syntax

this.<activity>.Context.TestProfile.**GetVariableValue**("<variable name>");

this.<activity>.Context.EnvironmentProfile.**GetVariableValue**("<variable name>");

Parameters

Parameter	Description
Variable name	A string for the name of the variable. You can find this value in the **Test Variables** tab 🌐 in the Properties pane.

Return Type

No explicit return - the variable value is set for the test run.

Examples

```
this.ConcatenateStringsActivity4.Context.TestProfile.GetVariableValue("Prefix");
this.ConcatenateStringsActivity4.Report("Prefix Variable Value", this.ConcatenateStringsActivity4.Context.TestProfile.GetVariableValue("Prefix"));
```

```
this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariableValue("Prefix");
this.ConcatenateStringsActivity4.Report("Prefix Variable Value 2", this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariableValue("Prefix"));
```

Import Method

Relevant for: API testing only

Description

Imports the specified Excel data source into your test.

Class

DataSource

Syntax

```
ExcelFileImportInputArgs <name> = new ExcelFileImportInputArgs(@"<path to data source>", "<data source name in the test>", "header row");
GetDataSource("<data source name>").Import (name);
```

> **Note:** You must cast the data source in the first line of the syntax in order to use the Import method. If you do not want to cast the data source, use the **ImportfromExcelFile** method instead.

Parameters

Parameter	Description
Name	Name assigned to the data source.
Path to data source	The Windows path to the file for the data source.
Data source name in the test	The name the test gives the data source after importing.
Header row	A boolean value if the data source has a header row. Possible values are "true" or "false".

Return Type

An Excel data source added to your test.

Example

```
ExcelFileImportInputArgs a = new ExcelFileImportInputArgs
```

```
(@"C:\Users\user\Documents\Unified Functional Testing\API Test Resources\ConcatenateStrings.xlsx", "Sheet1", true);
    GetDataSource("ConcatenateStrings!Sheet1").Import(a);
```

ImportFromExcelFile Method

Relevant for: API testing only

Description

Imports the selected Excel file to your test.

Class

DataSource

Syntax

GetDataSource("<data source name>").**ImportFromExcelFile**(@"<path to Excel file>", "<test data source name>", header row>);

Parameters

Parameter	Description
Name	Name assigned to the data source.
Path to data source	The Windows path to the file for the data source.
Data source name in the test	The name the test gives the data source after importing.
Header row	A boolean value if the data source has a header row. Possible values are "true" or "false".

Return Type

Adds an Excel data source to your test.

Example

```
GetDataSource("ConcatenateStrings!Sheet1").ImportFromExcelFile(@"C:\Users\user\Documents\Unified Functional Testing\API Test Resources\ConcatenateStrings.xlsx", "Sheet1", true);
```

Info Method

Relevant for: API testing only

Description

Reports the selected information to the UserLogger build log in the Output pane.

> **Note:** You can also use other common .NET logging options. For details, see http://www.codeproject.com/Articles/140911/log4net-Tutorial.

Class

ILog

Syntax

this.<activity>.Context.UserLogger.**Info**(message)

> **Note:** This method must always be used with a **UserLogger** object.

Parameters

Parameter	Description
Message	A string or value to report. You can use other code strings in this parameter to report the run-time value of any object, property, parameter, or variable in run-time.

Return Type

A message displayed in the UserLogger build log in the output pane.

Example

```
this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariablesNames().ToString();
var Variablenames = this.ConcatenateStringsActivity4.Context.EnvironmentProfile.GetVariables
Names();
Context.UserLogger.Info(Variablenames);
```

Report Method

Relevant for: API testing only

Description

Reports a custom attribute in the Captured Data pane of the run results.

Class

Action

Syntax

this.<activity>.**Report**("report attribute string", "<attribute value>");

Parameters

Parameter	Description
Report attribute string	A string for the custom field to enter into the run results.
Attribute value	The value for the attribute to report.

Return Type

A custom field in the run results.

Example

```
var ActivityArguments = args.Activity.StepId.ToString();
this.ConcatenateStringsActivity4.Report("Overall report", ActivityArguments);
```

SelectSingleNode Method

Relevant for: API testing only

Description

Selects a single node from an array containing input/output properties.

Class

This method is not part of a UFT API testing class, but is invoked by the **InputEnvelope** object.

Syntax

<activity>.InputEnvelope.**SelectSingleNode**("<fully qualified Xpath>").<supported object/method>

Parameters

Parameter	Description
Fully qualified XPath	The XPath to the property node in the array. Right-click the property/parameter name in the **Input/Checkpoints** tab in the Properties pane and select **Copy Fully Qualified XPath**to retrieve this information.

Return Type

No explicit return - enables the test to access the property or parameter.

Example

```
var GetFlights_Input_Departure_City = GetDataSource("WebServiceData!Input").GetValue(this.
Loop2.CurrentIterationNumber-1, "DepartureCity").ToString();
StServiceCallActivity4.InputEnvelope.SelectSingleNode("/*[local-name(.)='Envelope'][1]/*[local-n
ame(.)='Body'][1]/*[local-name(.)='GetFlights'][1]/*[local-name(.)='DepartureCity'][1]").InnerText
= GetFlights_Input_Departure_City;
```

SetValue Method

Relevant for: API testing only

Description

Sets the value of a specified item in the data source.

> **Notes:**
>
> - This method must be used with the **GetDataSource** method.
> - This method cannot be used for XML data sources.

Class

DataSource

Syntax

```
GetDataSource(<data source name>).SetValue(<row index>, "<column name>");
```

Parameters

Parameter	Description
Row index	**Required.** The row from which to take the value. **Note:** The row index is zero-based, meaning if you want to pull from the first row of the table, you must set the row index value to 0.
Column name	**Required.** The name of the column from which to take the value.

Return Type

No explicit return object. Sets a value in the specified data source.

Example

```
GetDataSource("ConcActivity Strings").SetValue(1, "Suffix", "done.");
```

SetVariableValue Method

Relevant for: API testing only

Description

Sets the value of a selected test or environment variable.

Class

TestProfile

Syntax

```
this.<activity>.Context.TestProfile.SetVariableValue("<variable name>" , "<variable value>");
```

```
this.<activity>.Context.EnvironmentProfile.SetVariableValue("<variable name>" , "<variable value>");
```

Parameters

Parameter	Description
Variable name	A string for the name of the variable. You can find this value in the **Test Variables** tab 🌐 in the Properties pane.
Variable value	The value for the variable. The format differs on the expected use of the variable.

Return Type

No explicit return - the variable value is set for use in the test run.

Example

```
var activity = ((HP.ST.Ext.WebServicesActivities.StServiceCallActivity)(args.Activity));
activity.Report( "codeCheckPointEvent" , "code CheckPoint event" );
args.Checkpoint.Report("CheckPoint","CheckPoint","=" , HP.ST.Fwk.RunTimeFWK.CheckpointFWK.StatusEnum.Succeed );
activity.Context.EnvironmentProfile.SetVariableValue( "codeCheckpointEvent","true");
```

Troubleshooting and Limitations - Event Coding

Relevant for: API testing only

This section describes troubleshooting and limitations for working with UFT events.

- The **Loop.CurrentIterationObject** is deprecated and will always return null.
- When calling a test or action, you cannot access the data associated with the called test or action using runtime API coding, even if overriding the data was enabled (through the **Allow other tools to override the data** option).
- When working on non-English operating systems, you must install a localized version of .NET framework to obtain localized compilation errors.
- By default, on machines running 64-bit Windows, the ActiveX object, **HP.ST.Test**, is only registered in 32-bit mode - not 64-bit.

  Workaround: Register the assembly manually using the following command line code:

  ```
  cd "<UFT installation folder>\bin" %windir%\
  Microsoft.NET\Framework64\v4.0.30319\
  RegAsm.exe HP.ST.Fwk.SOAReplayAPI.dll /codebase
  ```

Chapter 63: API Testing Extensibility

Relevant for: API testing only

The HP UFT installation includes the capabilities to create custom activities for the **Toolbox** pane. After defining a custom activity, you can drag it onto the canvas as you would with any other built-in activity.

This chapter includes:

Concepts

Creating New Activities - Overview

Relevant for: API testing only

UFT enables you to create custom API testing activities to extend the capabilities of the product.

Once you create a new activity through this mechanism, it will be available in the **Toolbox** pane for all future tests.

For most custom activities, you can use the Activity Wizard. For C# users, the wizard creates a Visual Studio project into which you can add your own C# code. Java users can edit .java files which will be compiled into .class files. You then deploy the new activity into UFT. For details, see "Activity Wizard" on page 2049 and "How to Use the Wizard to Create a Custom Activity - C#" on page 2036.

Advanced users can build custom activities manually, without the wizard. For details, see "How to Manually Create a Custom Activity in C#" on page 2040.

Custom Activity Files

Relevant for: API testing only

This section describes the structure and content of the files required to manually define a new activity in UFT. The following information is not relevant if you are using the Activity wizard.

To create a custom activity, you need to define the following files on all machines upon which you intend to run the test.

- "Runtime Files"
- "Signature Files"
- "Addin Files"

The **Runtime** file is the DLL that UFT invokes to run the activity. For details, see "Runtime Files" on the next page.

The **Signature** file is an XML file that defines the input and output properties, events, and the runtime class that executes the activity. For details, see "Signature Files" on page 2025.

The **Addin** file is an XML file that references all of the activity component data. For details, see "Addin Files" on page 2032.

In addition, you can also define resource files to store text strings used by your activity. For details, see "Resource Files " on page 2035.

The product's installation includes a sample project in the ExtensibilitySamples folder. Use this sample as a basis for a new activity.

All of the custom files—Signature, Addin, and Runtime—should be stored in the <Installation_folder>\addins\CustomerAddins\<addin_name> folder. This enables UFT to load them during startup.

For more details, see "How to Manually Create a Custom Activity in C#" on page 2040.

> **Note:** This is a preliminary version of the SDK (Software Development Kit). It enables you to extend the capabilities of the product. However, this SDK is subject to change in a future release, and these changes might require you to update any code that uses this preliminary version. Although HP endeavors to keep these changes to a minimum, we cannot guarantee that extensions created using the preliminary version of the SDK will continue to work without modification when upgrading to a new version of HP UFT.

Runtime Files

Relevant for: API testing only

In addition to the signature and addin files, you must provide a DLL to run when executing the activity. You create a solution and customize the code as required. You can use the sample ReportMessageActivitySample.sln located in the product's ExtensibilitySamples folder as a basis for your project.

Use Microsoft's IntelliSense to determine the method's arguments and syntax.

Here are some guidelines that you must follow:

- As shown in the sample, you must use methods that are included in the **STActivityBase** class.

  ```
  public class ReportMessageActivitySample : STActivityBase
  ```

- **STExecutionResult** is the return value of the **ExecuteStep** method. It receives one or two parameters: **STExecutionStatus Status** and optionally, **string msg**. STExecutionStatus is an **enum** type that can be set to the following values: ActivityFailure, ActivityStopTest, ApplicationUnderTestFailure, Equals, ReferenceEquals, Success, TestStopped. In the following example it receives the **Success** value.

  ```
  return new STExecutionResult(STExecutionStatus.Success);
  ```

- Place your executable code within the **ExecuteStep** function.

  ```
  protected override STExecutionResult ExecuteStep()
    { ...
  ```

  > **Note:** You must compile the DLL with a **Target Framework** of Framework 4.0, available in Microsoft Visual Studio 10.

For task details and an example, see "How to Create a Runtime File" on page 2044.

Signature Files

Relevant for: API testing only

The signature file describes the activity to UFT. It typically has a **Resource** element followed by the following sections: **GeneralProperties**, **InputProperties**, **OutputProperties**, **Tabs**, and **Events**. The signature file must have an .xml extension.

This following sections describe the elements of the signature file:

Resource Element

Relevant for: API testing only

The **Resource** element has the following attributes:

Attribute	Description
type	The type of entity, in this case **Activity**.
id	A unique string that identifies the activity. You can reference this ID when writing event handler code. For details, see the section on API coding.
version	The version of the current addin mechanism. For example, 1.0.0
group	The parent group under which the activity will appear in the **Toolbox** pane, for example **String Manipulation**. To add it to an existing group, specify a group name as it appears in the **Toolbox** pane. If the group does not exist, it adds a new group.
shortname	The short name of the activity as shown in the **Toolbox** pane's hint area (above the description).
description	The description of the activity as shown in the **Toolbox** pane's hint area.
assembly	The DLL file to call when running the activity, stored in the same folder as the signature and addin files.
className	The class implemented by the activity. **Note:** The class must inherit from the STActivityBase class.

Attribute	Description
image	An image file for the icon representing the activity. Store the image in the same folder as the signature file.
visible	A boolean value indicating whether to display the activity is displayed in the **Toolbox** pane.
xmlns	The namespace that defines the schema for the signature file, http://hp.vtd.schemas/signature/v1.0. Keep the default value.
xmlns:xsi	The schema instance used for the signature file. Keep the default value, http://www.w3.org/2001/XMLSchema-instance.
xmlns:Location	The URL of the signature file's schema, **Signature.xsd**, referenced by the namespace. This value has two parts: the namespace and the file path. • **Namespace**. http://hp.vtd.schemas/signature/v1.0 • **File path**. <Installation_Folder>/dat/schemas/Signature.xsd Keep the default value.

The following example shows the **Resource** section from the signature file of the sample activity, **ReportMessageActivitySample**. The location of the sample is: <*Installation_Folder*>\ExtensibilitySamples\ReportMessageActivitySample

```
<Resource
  type="Activity"
  id="ReportMessageActivitySample"
  version="1.0.0"
  group="Miscellaneous"
  shortName="ReportMessageActivitySample"
  description="The ReportMessageActivitySample allows you to send a custom message to the re
port and/or log. "
  assembly="ReportMessageActivitySample.dll"
  className="ReportMessageActivitySample.ReportMessageActivitySample"
  image="toolbox_ReportMessageActivitySample.png"
  visible="true"
  xmlns="http://hp.st.schemas/signature/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://hp.st.schemas/signature/v1.0 ../../../dat/schemas/Signature.xsd"
  >
```

Section Element

Relevant for: API testing only

The Section definitions apply to all sections in the Signature file, such as **GeneralProperties**, **InputProperties**, and **OutputProperties**. The following table describes the primary **Section** element attributes.

Attribute	Description
name	The internal name of the section. To use a sub-element, it is recommended that you set the value to the name of the sub-element. For example name="Tab" or name="Alerts".
source	If set to **true**, it displays the source of the section.
dest	**destination**. If set to **true**, it displays the destination path of the section.
checkpoint	If set to **true**, displays the checkpoint check box in the **Validate** column.
isSharedMetadata	When **true**, enables the section to share its meta data with other sections.
propertiesType	The type of the properties in the section, for example, "XML".
showXmlControls	When **true**, displays the XML controls such as Text and XPath tabs in the section.
displayName	The name of the section as it will appear in the Properties pane.

The following table describes the sub-elements of **Section**:

Element	Description
Tab	The tabs to appear in the Properties pane using the following attributes. For details, see below. • **name**. The internal name of the tab. Some of the built-in ones are General, InputOutput, Events, Attachments, and SOAPFault. • **id.** The id of the tab referred to be the API. The id usually uses the name with an added suffix, "Tab". For example, GeneralTab, InputOutputTab, and EventsTab. • **CanBeInToolbox.** If true, shows the tab on the **Toolbox** pane's toolbar. • **CanBeInPropertySheet.** If true, shows the tab in the Properties pane. • **CanBeInDataLinkDialog.** If true, shows the tab in the "Select Link Source Dialog Box (API Testing)" (described on page 1840). **Note:** To use the default tabs: **General**, **Input/Checkpoints**, and **Events**, you do not need to include this element. If you want to omit one of the tabs or add extra ones, then you need to include the Tabs sub-element and specify the desired tabs. For details about the tabs, see "Properties Pane" on page 385.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td><b>Alert</b></td><td>Enables you to apply alerts to the properties in the section using the following attributes:
<ul>
<li><b>constraint.</b> The reason to show the alert, for example, NullValueConstraint.</li>
<li><b>target.</b> The Xpath of the property to which to apply the constraint.</li>
<li><b>section.</b> The internal name of the section containing the properties.</li>
<li><b>type.</b> The type of alert, such as error or warning</li>
</ul>
The value of the element, is the alert message. For example:
<pre>
<Section name="Alerts" isSharedMetaData="true">
  <Alert constraint="NullValueConstraint"
        target="/Url[1]"
        section="InputProperties"
        type="error">The 'URL' must not be empty</Alert>
</Section>
</pre></td></tr>
</table>

Element	Description
Events	In the **Events** sub-element, you can customize the events that will be available for the activity. This sub-element uses the following attributes • **name.** The internal name of the event. Use one of the built-in names or define a custom one. ▪ **CodeCheckpointEvent.** Enables you to create an event handler to run when the test is verifying checkpoints. ▪ **BeforeExecuteStepEvent.** Enables you to create an event handler to run before executing the activity. ▪ **AfterExecuteStepEvent.** Enables you to create an event to run after executing the activity. ▪ **<custom event>.** A custom event that you define. • **description.** A textual description of the event. • **eventArgs.** The source of the arguments for the event. The standard argument for **BeforeExecuteStepEvent** and **AfterExecuteStepEvent** event is STActivityBaseEventArgs. The built-in value for the **CodeCheckpointEvent** is CheckpointEventArgs. Web Service steps provide additional built-in events. For details, see "Updating Web Services" on page 1854. **Note:** To access the default events: **CodeCheckpoint**, **BeforeExecute**, and **AfterExecute**, you need to include only the **Events** tab in the **Tab** sub-element, but you do not need to use the **Events** sub-element. If you want to omit one of the events or add custom events, then you need to include this sub-element and specify the desired events.

Property Definitions

Relevant for: API testing only

The property definitions in the signature file, apply to all sections that use properties. The built-in sections that use properties are:

- **GeneralProperties.** Defines the properties in the **General** view of the Properties pane, for example **Step ID** and **Name**.
- **InputProperties.** Defines the input properties located in the Input pane of the of the Properties pane's **Input/Checkpoints** tab.
- **OutputProperties.** Defines the output properties located in the **Checkpoints** pane of the of the Properties pane's **Input/Checkpoints** tab.

This section includes:

- "Elements and Sub-Elements" below
- "Element Attributes" on the next page
- "Simple Elements with Enumeration" on page 2032
- "Complex Array Elements" on page 2032

Elements and Sub-Elements

The elements and attributes are defined in the standard XML schema file, http://www.w3.org/2001/XMLSchema, or the built-in types.xsd schema, located in the <Installation_Folder>/dat/schema folder. The following table describes the elements and sub-elements that can be used in these sections. The level number indicates the level of the element or sub-element in the hierarchy.

Element	Description
xs:schema	The schema namespaces for the properties, as described by the **xml:ns** attribute. Keep the default values, http://hp.vtd.schemas/types/v1.0 and http://www.w3.org/2001/XMLSchema.
xs:import	The namespace to import using the **namespace** and **schemaLocation** attributes. Keep the default values, http://hp.vtd.schemas/types/v1.0 and ../../../dat/schemas/types.xsd.
xs:element	The element to define, using the attributes described in the table below.

Element	Description
xs:simpleType	A tag indicating the beginning of definitions of a simple type property. **Note:** You only need to enclose a simple type element with this tag, if you want to do enumeration with a drop-down list. For example, the following definition does not require an **xs:simpleType** tag. <xs:element name="ClientCertificate" type="types:Certificate" types:displayName="Client certificate" />
xs:complexType	A tag indicating the beginning of definitions for a node of multiple properties.
xs:sequence	A tag indicating the beginning of a list of properties in a complex type property.
xs:restriction	A tag restricting the value of the enumeration values of a property, using the **base** attribute. To restrict **String** type values, use base="xs:string".
xs:enumeration	A tag indicating the beginning of list of values in the drop-down list for a property, using the **value** attribute.
xs:annotation	An annotation for the element Use an **xs:documentation** sub-element to compose text that will appear below the properties grid in the Properties pane.

Element Attributes

The following table describes the primary attributes of the **xs:element**. For attributes in the standard XML schema, use an **xs:** prefix in the value, for example standard types use type=xs:string or type=xs:int.

For types defined in the **Types.xsd** schema, use a **types:** prefix in the attribute name. For example types:displayName.

Attribute	Description
name	The internal name of the property or grid in the Properties pane. This is the name referenced by other calls and by the event handlers code. This is not the name displayed in the Properties pane's **Name** column.
type	The type of property. Some common values are: xs:string, xs:int, xs:boolean, Multipart, Header, Part. For a value defined in the Types schema, use the **types**: prefix. For example type="types:filePath".
minOccurs	The minimum number of array elements for which the user must provide. For none, specify "**0**".

Attribute	Description
maxOccurs	The maximum number of array elements the user may provide. To allow an unlimited amount, specify "**unbounded**"
types:visible	When **true**, enables the parameter to be visible even before being expanded by the Add Array Element command.
types:argType	The type of the property: "XML" or "Object".
types:displayName	The property name as it will appear in the Properties pane.

Simple Elements with Enumeration

The following **ReportMessageActivitySample** example defines an input parameter, **Status,** with an enumeration attribute. This code creates a drop-down list of values in the Properties pane's input property grid.

```
<xs:element name="Status" default="Done" types:displayName="Status">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:enumeration value="Done"/>
         <xs:enumeration value="Pass"/>
         <xs:enumeration value="Fail"/>
        </xs:restriction>
       </xs:simpleType>
</xs:element>
```

Complex Array Elements

The following sample defines a complex property, with a Key and Value pair of values.

```
<xs:complexType name="NameValueType">
  <xs:sequence>
    <xs:element name="Key" type="xs:string"/>
    <xs:element name="Value" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Addin Files

Relevant for: API testing only

The addin file provides the references for the activity you are defining. The file is in XML format and contains information such as activity names, dependencies, and runtime DLLs.

The addin file should be located in the installation directory under the addins\CustomerAddins\<addin_name> folder, together with the signature file. The addin file must have an .addin extension.

Each addin file should contain the following sections:

- "Addin Section" below
- "Manifest Section" below
- "Runtime Section" on the next page

Addin Section

The Addin's attributes describe the activity to UFT.

Element	Description	Attributes
<Addin>	Basic details about the addin.	• **name.** the name of the addin. • **author.** the creator of the activity. • **copyright.** the full path of a text file with the copyright information. • **description.** a textual description of the activity. • **version**. The addin file version, set to 1.0.
Path	Details about the location of the activity.	• **name.** The logical path scanned by the framework, to identify addins. The physical location of this folder is addins\CustomerAddins\<addin_name>.
Activity (sub-element of Path)	See attribute descriptions.	• **id.** An identifying string corresponding to the ID in the signature file. • **displayName.** The activity's display name in the **Toolbox** pane. • **signatureFile.** The name of the XML signature file.

Manifest Section

The Manifest section contains a unique logical name for the addin and provides a list of dependencies.

Element	Description	Attributes
Identity	Basic details about the addin.	• **name.** The activity name corresponding to the **ID** in the signature file. When referring to this addin as a dependency, use this name.

Element	Description	Attributes
Dependency	The activity upon which the current activity is dependent	- **addin**. The identity name of the dependent activity, from the name attribute in the **Manifest** section of its addin file. - **requirePreload**. A boolean value indicating whether to preload the dependent addin before loading the current one.

Runtime Section

The runtime section contains information about the addin's runtime file.

Element	Description	Attributes
Import	An assembly to import when running the activity.	- **assembly**. The name of an assembly. Use the DLL name without the DLL extension. **Note:** To import an addin from another activity, precede the addin name with a colon. For example, :HP.ST.Fwk.DesignerModel imports the DesignerModel addin.

The following example shows the **ReportMessageActivitySample.addin** file. For multiple activities, use unique **Addin** files.

```
<?xml version="1.0" encoding="utf-8"?>
<AddIn name       = "HP Report Message Activity Sample"
  author     = "John Doe"
  copyright   = "C:\Copyrights\copyright.txt"
  description = "Extensibility Sample - Report Message Activity"
  version="1.0">
 <Manifest>
  <!--<Must be unique -->
  <Identity name = "ReportMessageActivitySample"/>
 </Manifest>
 <Runtime>
  <Import assembly=":HP.ST.Fwk.DesignerModel"/>
 </Runtime>
<Path name = "/ST/Activities">
  <!--Misc Activities -->
  <Activity id    = "ReportMessageActivitySample"
     displayName   = "ReportMessageSample"
     signatureFile = "ReportMessageActivitySample.xml"
     assembly="ReportMessageActivitySample.dll"/>
 </Path>
```

```
</AddIn>
```

Resource Files

Relevant for: API testing only

You can use resource files to retrieve values for elements and attributes. The **fromResource** function lets you name the resource containing the values.

In the following example, the signature file retrieves the **shortName** and **description** from a resource file.

```
<Resource
 type="Activity"
 id="ReportMessageActivitySample"
 version="1.0.0"
 group="Miscellaneous"
 shortName="fromResource(conc_str_short_name)"
 description="fromResource(conc_str_description)"
```

The resources are defined in a standard Microsoft ResX Schema version 2.0 Resource fie.

```
<data name="conc_str_description" xml:space="preserve">
   <value>Reports an activity's run status to the log</value>
 </data>
 <data name="conc_str_short_name" xml:space="preserve">
   <value>Report Message</value>
 </data>
```

The resource reference must be a compiled file with a **.resources** extension, compiled from the ResX source file and stored in the same folder as the signature file.

You can generate the compiled file as a post-build operation using the **resgen** utility. For example:

```
resgen STBasicActivity.resx STBasicActivity.resources.
```

For step-by-step instructions on how to create a custom activity, see "Prerequisite - create a runtime file" on page 2041.

Tasks

How to Use the Wizard to Create a Custom Activity - C#

Relevant for: API testing only

This task describes how to create a new activity, using C#, and deploying it in UFT.

This task includes the following steps:

- "Run the Activity Wizard" below
- "Open the folder" below
- "Add execution code" below
- "Add Logger code - optional" on the next page
- "Add a Report statement - optional" on the next page
- "Compile the project into a DLL" on the next page
- "Deploy the activity" on the next page
- "Add a test step" on the next page

1. Run the Activity Wizard

Open the Activity Wizard from the product's start menu (**Start > All Programs > HP Software > HP Unified Functional Testing> Tools > Activity Wizard**).

In the wizard's **General Properties** pane, select the **C#** as the **Language**.

Define the relevant properties and proceed to the last screen of the wizard. For details, see the "Activity Wizard" on page 2049.

> **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

2. Open the folder

On the final screen of the wizard, click **Open Folder** to open the <Activity Name> folder, corresponding to the activity name you specified in the wizard. Navigate to the following subfolder SourceCode and locate the <Activity Name>.cs file.

3. Add execution code

Add your execution code to the **ExecuteStep** function inside the .cs file.

```
protected override STExecutionResult ExecuteStep()
{
try
{
//**************************
// Execution code goes here  //**************************
...
```

4. **Add Logger code - optional**

   Add information for the log using the **LogInfo**, **LogDebug,** or **LogError** statements. For example:

```
protected override STExecutionResult ExecuteStep()
{
try
{
   LogInfo("Log Message 1");
   LogDebug("Log Message 2");
   LogError("Log Message 3");
...
```

5. **Add a Report statement - optional**

   Add a **Report** statement. For example:

```
protected override STExecutionResult ExecuteStep()
{
try
{
      DetailsReport = DetailsReport.Replace("\\n", "<BR>");
      this.Report("Message", DetailsReport);    ;
...
```

6. **Compile the project into a DLL**

   Build the project and make sure the current <Activity Name> .dll file is in the new activity folder that you specified in the wizard.

7. **Deploy the activity**

   In the final wizard screen, click **Deploy in UFT**. Click **Finish** to close the wizard. Restart UFT.

8. **Add a test step**

- Open a new test and add the new custom activity (by default under the **Miscellaneous** category) into the Test Flow.
- Provide values for the properties you defined for the activity in the wizard.
- Run the test and observe the Output log and Run Results Viewer.
- Enable checkpoints to verify the results and rerun the test.

How to Use the Wizard to Create an Activity - Java

Relevant for: API testing only

This task describes how to create a new activity using Java code, and deploy it in UFT.

This task includes the following steps:

- "Prerequisite" below
- "Run the Activity Wizard" below
- "Open the folder" on the next page
- "Edit the code" on the next page
- "Add Logger code - optional" on the next page
- "Add a Report statement - optional" on the next page
- "Compile the Java into a class" on page 2040
- "Deploy the activity" on page 2040
- "Add a test step" on page 2040

1. **Prerequisite**

   Make sure you have a JAVA_HOME environment variable defined on your machine indicating the parent JDK folder.

2. **Run the Activity Wizard**

   Open the Activity Wizard from the product's start menu (**Start > All Programs > HP Software > HP Unified Functional Testing > Tools >Activity Wizard**).

   In the wizard's **General Properties** pane, select the **Java** as the **Language**.

Define the relevant properties and proceed to the last screen of the wizard. For details, see the "Activity Wizard" on page 2049.

> **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

3. **Open the folder**

   On the final screen of the wizard, click **Open Folder** to open the **<Activity Name>** folder, corresponding to the activity name you specified in the wizard. Navigate to the following subfolder <Activity Name>\hp\st\ext\java and locate the MyLogic.java file.

4. **Edit the code**

   Edit the ExecuteLogic function inside the MyLogic.java file. Make sure to keep the **Properties** definition.

```
public Properties Props = new Properties();
public ExecutionResult ExecuteLogic()
{
   try{
   //**************************
   // Execution code goes here
   //**************************
   return ExecutionResult.Success;
   }
...
```

5. **Add Logger code - optional**

   Add information for the log using the **Logger.LogInfo**, **Logger.LogDebug,** or **Logger.LogError** statements. For example:

```
   try{
   ...
      Logger.LogInfo("Log Message 1");
      Logger.LogDebug("Log Message 2");
      Logger.LogError("Log Message 3");
   ...
   return ExecutionResult.Success;
   }
```

6. **Add a Report statement - optional**

Add a Report statement, **Reporter.Report**, using key value combinations. For example:

```
try{
...
   Reporter.Report{"Name","John");
...
return ExecutionResult.Success;
}
```

7. **Compile the Java into a class**

   a. In your design IDE, add the ServiceTestCall.jar file to the build path.

   b. In UFT, run the CompileJavaFiles batch file in the <Activity Name>\hp\custom\java\activity folder to compile all java files into a class. This utility only compiles the files in its folder.

8. **Deploy the activity**

   In the final wizard screen, click **Deploy in UFT**. Click **Finish** to close the wizard. Restart UFT.

9. **Add a test step**

   - Open a new test and add the new custom activity (by default under the **Miscellaneous** category) into the Test Flow.
   - Provide property values in the Properties pane.
   - Run the test and observe the Output log and Run Results Viewer.
   - Enable checkpoints to verify the results and rerun the test.

How to Manually Create a Custom Activity in C#

Relevant for: API testing only

This task describes how to create a new activity and implement it into UFT.

To run a test with the custom activity on another machine, you need to copy all of the custom files to its <Installation_Folder>\addins\CustomerAddins\<addin_name> folder.

This task includes the following steps:

- "Prerequisite - create a runtime file" on the next page
- "Create a signature file" on the next page

- "Create an addin file" on the next page
- "Provide a graphic for your activity - optional" on page 2043
- "Check the implementation" on page 2043

1. **Prerequisite - create a runtime file**

   Create a C# project that implements your activity's actions in the addins\CustomerAddins\<addin_name> folder. For task details, see "How to Create a Runtime File" on page 2044.

2. **Create a signature file**

   a. Create a new signature file with an .xml extension. in the addins\CustomerAddins\<addin_name> folder, together with the runtime file. Use the sample project in the <installation folder>\ExtensibilitySamples folder as a basis for your custom signature file.

   b. Customize the **Resource** section or copy the code provided below, modifying the bolded text for your needs. For details about each of the elements, see "Resource Element " on page 2025.

```
<Resource
  type="Activity"
  id="ReportMessageActivitySample"
  version="1.0.0"
  group="Miscellaneous"
  shortName="ReportMessageActivitySample"
  description="ReportMessageActivitySample allows you to send a custom messa
ge to the report and/or log. "
  assembly="ReportMessageActivitySample.dll"
  className="ReportMessageActivitySample.ReportMessageActivitySample"
  image="toolbox_ReportMessageActivitySample.png"
  visible="true"
  xmlns="http://hp.st.schemas/signature/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://hp.st.schemas/signature/v1.0 ../../../dat/schemas/Signatur
e.xsd"
  >
```

   c. Add the required sections, such as GeneralProperties, InputProperties, Tabs, Events and so forth. For a list of the built-in sections and their attributes, see the "Section Element" on page 2026.

d. Add properties to the relevant sections. For details and examples, see "Property Definitions" on page 2030.

   - **GeneralProperties.** Properties displayed in the Properties pane's **General** tab. In most cases you can use the section as it appears in the sample file, without any modifications. By default, it will provide the **Step ID** and **Name** properties.

   - **InputProperties.** Properties displayed in the Properties pane's **Input/Checkpoints** tab, in the **Input** pane.

   - **OutputProperties.** Properties displayed in the Properties pane's **Input/Checkpoints** tab, in the **Checkpoints** pane.

e. Specify any external resource files as described in "Resource Files " on page 2035.

f. Close the file with the </Resource> tag.

```
</Resource>
```

For more details about the structure of the signature file, see "Signature Files" on page 2025.

3. **Create an addin file**

a. Create a new file with an .addin extension in the <installation directory>\addins\CustomerAddins\
<addin_name> folder, together with the signature file.

b. Use the sample addin file in the <installation folder>\ExtensibilitySamples folder as a basis, or copy the code provided below, modifying the bolded text for your needs. For details, see "Addin Files" on page 2032.

```
<?xml version="1.0" encoding="utf-8"?>
<AddIn name       = "HP Report Message Activity Sample"
      author      = "John Doe"
      copyright   = "prj:///doc/copyright.txt"
      description = "Extensibility Sample - Report Message Activity"
        version="1.0">
<Manifest>
   <!--<Must be unique -->
   <Identity name = "ReportMessageActivitySample"/>
 </Manifest>
 <Runtime>
   <Import assembly=":HP.ST.Fwk.DesignerModel"/>
 </Runtime>
 <Path name = "/ST/Activities">
  <!--Misc Activities -->
  <Activity id    = "ReportMessageActivitySample"
  displayName   = "ReportMessageSample"
   signatureFile = "ReportMessageActivitySample.xml"
  assembly="ReportMessageActivitySample.dll"/>
 </Path>
</AddIn>
```

c. Create a unique Addin file for each activity—do not define multiple activities in a single Addin file. For details, see "Addin Files" on page 2032.

d. Define post-build tasks such as resgen, as described in "Resource Files " on page 2035.

e. Compile the project and copy the DLL to the <Installation_Folder>\addins\CustomerAddins\<addin_name> folder.

For additional details, see "Runtime Files" on page 2024.

4. **Provide a graphic for your activity - optional**

a. Copy an icon image for your activity into the <Installation_Folder>\addins\CustomerAddins\<addin_name> folder. This file should meet the following requirements:

   - a .png extension
   - sized at 16 x 16 pixels
   - 8-bit color depth

b. Specify the name of the image file in the signature file's "Resource Element ".

5. **Check the implementation**

a. Reopen the application and drag the new activity into the Test Flow. Verify that the activity and its properties appear as expected.

b. Provide property values.

c. Run the test and observe the Output log and Run Results Viewer.

d. Enable checkpoints to verify the results and rerun the test.

How to Create a Runtime File

Relevant for: API testing only

This section contains the following topics:

- "Add Using statements" below
- "Specify the namespace and class" below
- "Set the internal logging" on the next page
- "Initialize the properties" on the next page
- "Retrieve the property values" on the next page
- "Define events" on page 2046
- "Execute the step" on page 2047
- "Set the status" on page 2047
- "Compile the runtime file" on page 2048

Note: You must compile the DLL with a **Target Framework** of Framework 4.0.

1. Add Using statements

Provide the mandatory **using** statements. In your solution, you must also add a reference to the .dll files. The .dll files are located in the products installation's /bin folder. You must always add a reference to HP.ST.Fwk.RunTimeFWK.dll. If you are using internal logging, you must also add a reference to log4net.dll.

The following example shows the **Using** statements in the sample .cs file.

```
using HP.ST.Fwk.RunTimeFWK;
// If you need to implement Internal Logging
using log4net;
```

2. Specify the namespace and class

Define the namespace and provide the activity's runtime code. The class you define for your custom activity must inherit from the STActivityBase class. For example:

```
namespace ReportMessageActivitySample
```

```
{
[Serializable]
public class ReportMessageActivitySample : STActivityBase
{
```

3. Set the internal logging

Use the built-in logger manager to instruct the activity to create an internal log during runtime. This example gets the property values of the input properties and sends the output to either the Run Results Viewer only or to the Run Results Viewer and Output window. For example:

```
/// <summary> /// Internal log
/// </summary>
private static readonly ILog log =
    LogManager.GetLogger(typeof(ReportMessageActivitySample));
const string runResults = "Run Results";
const string runResultsAndOutputWindow = "Run Results and Output Window";
```

For details about other logging options, see "Assert Object" on page 1989.

4. Initialize the properties

Initialize the custom Input and Output properties that you define in the signature file. The following example initializes the three input properties: **Status**, **Message**, and **Destination**. For example:

```
/// <summary>
/// Initializes properties.
/// </summary>
/// <param name="ctx">The runtime context</param>
public ReportMessageActivitySample(ISTRunTimeContext ctx, string name)
: base(ctx, name)
{
this.Status = String.Empty;
this.Message = String.Empty;
this.Destination = String.Empty;
}
```

5. Retrieve the property values

This section retrieves or sets the input property values. For example:

```
/// <summary>
/// Gets or sets the status of the message to report.
/// </summary>
public string Status { get; set; }
/// <summary>
/// Gets or sets the details of the message to report.
/// </summary>
public string Message { get; set; }
/// <summary>
/// Gets or sets the destination where the message should be reported to.
/// </summary>
public string Destination { get; set; }
```

If you have array type properties that are not described by a schema, for example, key/value pairs, you must initialize all the members of the array explicitly, and indicate the actual number of elements.

The following example initializes 40 elements for the MyArrayName property. It contains 40 key and value pairs.

```
this. MyArrayName = new MyPair[40];
for (int i=0; i<40; i++)
{
this. MyArrayName [i] = new MyPair();
}
public MyPair[] MyArrayName;
public class MyPair
{
string Key;
string Value;
}
```

For arrays defined by a schema or WSDL, you can use the standard "Select Link Source Dialog Box (API Testing)" (described on page 1840) and link directly to the array element.

6. Define events

Define one or more custom events, that you will invoke later. For example:

```
public event EventHandler CustomerEvent;
private void InvokeCustomerEvent(EventArgs MyArg)
{
EventHandler handler = this.CustomerEvent;
if (handler != null)
{
handler(this, MyArg);
}
}
```

7. **Execute the step**

   Execute the step and send the runtime information to the log. Use the **STExecutionResult** data type and its **ExecuteStep** function defined in the **STActivityBase** class. For example:

```
protected override STExecutionResult ExecuteStep()
{
string DetailsReport;
if (this.Destination == runResultsAndOutputWindow)
{
LogInfo("\n" + this.Message.Replace("\\n", "\n"));
}
/// <summary>
/// Reports message to test results and output window.
/// </summary>
// The line-breaks replacements allow the printing of multiple lines in the report
    DetailsReport = this.Message;
    DetailsReport = DetailsReport.Replace("\\n", "<BR>");
    DetailsReport = DetailsReport.Replace("\n", "<BR>");
    this.Report("Message", DetailsReport);
```

   If you defined a custom event, invoke it after the call to ExecuteStep.

```
...
protected override STExecutionResult ExecuteStep()
{
InvokeCustomerEvent();
}
```

8. **Set the status**

   Set the Status of the test run. The ReportMessageActivitySample.cs sample file uses enumeration to set the status, based on the **STExecutionResult** value. For example:

```
switch (this.Status)
{
case "Done":
this.Report(ReportKeywords.StatusKeywordTag, ReportKeywords.DoneValueTag);
return new STExecutionResult(STExecutionStatus.Success);
case "Pass":
```

```
this.Report(ReportKeywords.StatusKeywordTag, ReportKeywords.SuccessValueTag);
return new STExecutionResult(STExecutionStatus.Success);
case "Fail":
this.Report(ReportKeywords.StatusKeywordTag, ReportKeywords.FailureValueTag);
return new STExecutionResult(STExecutionStatus.Success);
default:
return new STExecutionResult(STExecutionStatus.Success);
}
```

9. **Compile the runtime file**

   After you customize the code, you compile the .dll. The .dll name should be the same as the name of the addin file. For example, the runtime file, ReportMessageActivitySample.dll corresponds to the ReportMessageActivitySample.addin file.

   After you create the runtime file in your development environment, you reference the .dll from the signature file, and the signature file from the addin file.

Reference

Activity Wizard

Relevant for: API testing only

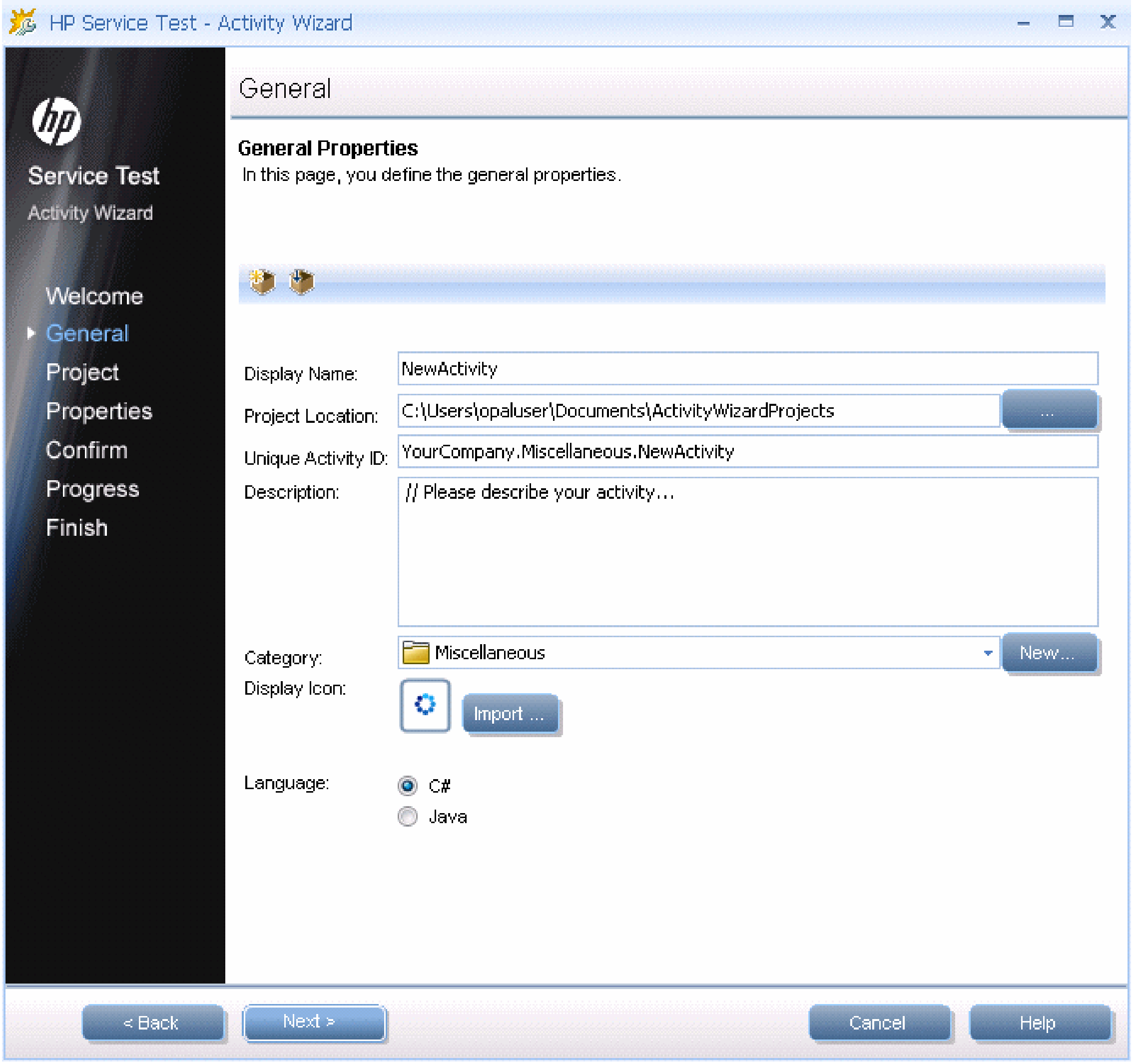
This wizard enables you to create new custom activities in the **Toolbox** pane.

To access	**Start > All > Programs > HP Software > HP Unified Functional Testing > Tools > Activity Wizard** **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.
Wizard map	This wizard contains: **Welcome** > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "
See also	• "Creating New Activities - Overview " on page 2023 • "How to Manually Create a Custom Activity in C#" on page 2040 • "How to Use the Wizard to Create a Custom Activity - C#" on page 2036

General Properties Page

Relevant for: API testing only

This wizard page enables you to set the General type properties for the activity.



General

General Properties

In this page, you define the general properties.

Display Name: NewActivity

Project Location: C:\Users\opaluser\Documents\ActivityWizardProjects

Unique Activity ID: YourCompany.Miscellaneous.NewActivity

Description: // Please describe your activity...

Category: Miscellaneous

Display Icon:

Language: ● C# ○ Java

Important information	General information about this wizard is available here: "Activity Wizard" on the previous page.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

User interface elements are described below:

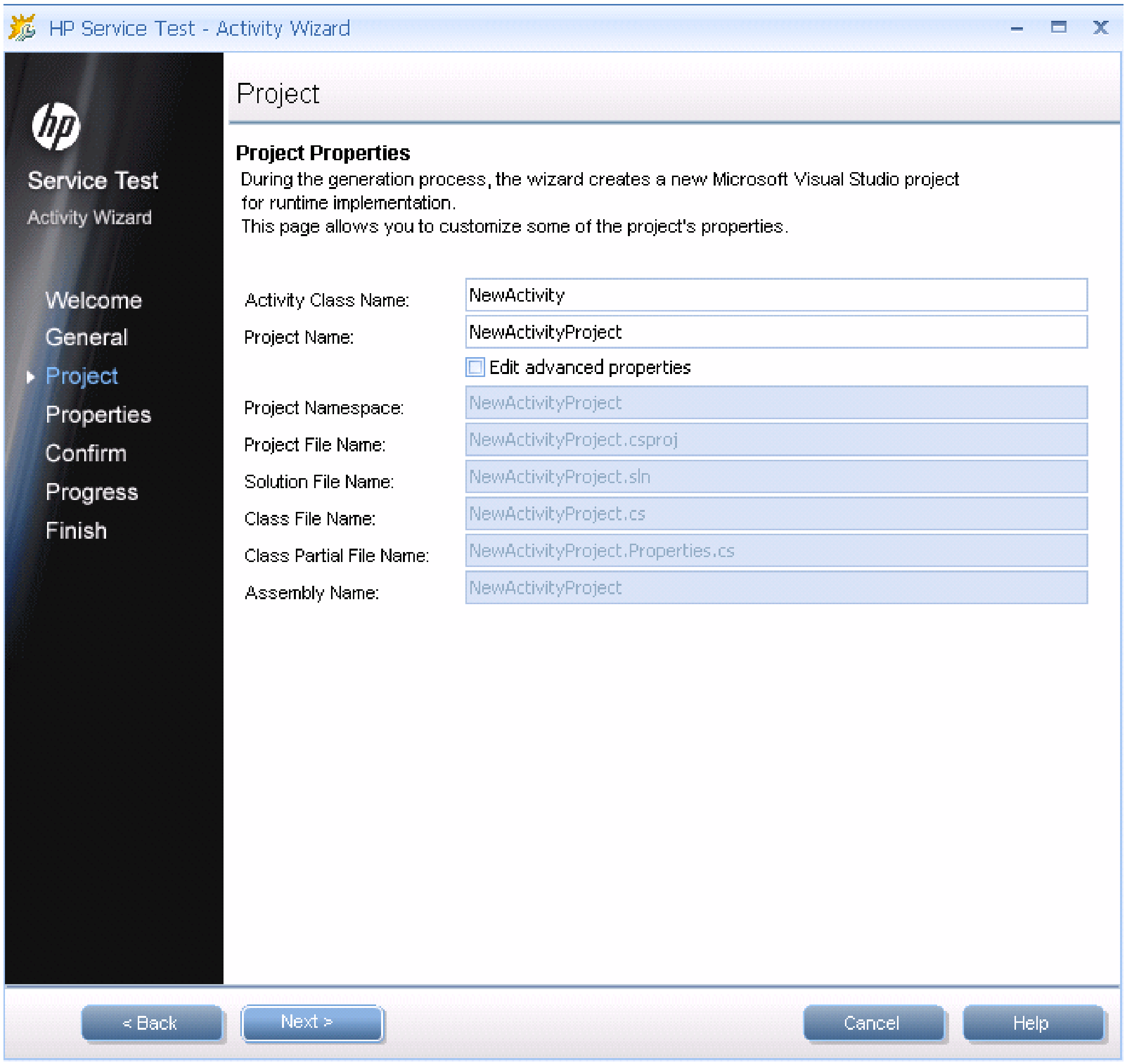
UI Elements	Description
📦	**Reset Values.** Resets all of the values to their original defaults and discards all changes that you made.

UI Elements	Description
📦	**Open Wizard Project.** Opens a Visual Studio activity project that you created earlier.
Category	The category under which to deploy the activity. A drop down list contains all of the built-in or previously created categories. **Clicking on New** opens the Add New Category dialog box.
Description	A meaningful description that will appear in the **Toolbox** pane's hint area when you select the activity.
Display Icon	An icon image for the activity for the **Toolbox** pane and canvas. Click **Import** to upload a new image file for the icon. The icon size should be 16 x 16 pixels.
Display Name	The name of the activity as it will appear in the canvas and **Toolbox** pane.
Language	The language in which to generate the code: C# or Java. **Note:** When you select **Java**, the wizard creates a Java file to which you can add the logic for your activity, and references to other required files. In addition, a Visual Studio project is created to manage the integration between the API Test and your Java activity.
Project Location	The location in which to save the project.
Unique Activity ID	A unique ID for the new activity. This ID can be used to reference the activity.

Project Properties Page

Relevant for: API testing only

This wizard page enables you to set the properties for the activity project.



Project

Project Properties

During the generation process, the wizard creates a new Microsoft Visual Studio project for runtime implementation.
This page allows you to customize some of the project's properties.

Activity Class Name: NewActivity
Project Name: NewActivityProject
□ Edit advanced properties
Project Namespace: NewActivityProject
Project File Name: NewActivityProject.csproj
Solution File Name: NewActivityProject.sln
Class File Name: NewActivityProject.cs
Class Partial File Name: NewActivityProject.Properties.cs
Assembly Name: NewActivityProject

Important information	General information about this wizard is available here: "Activity Wizard" on page 2049.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

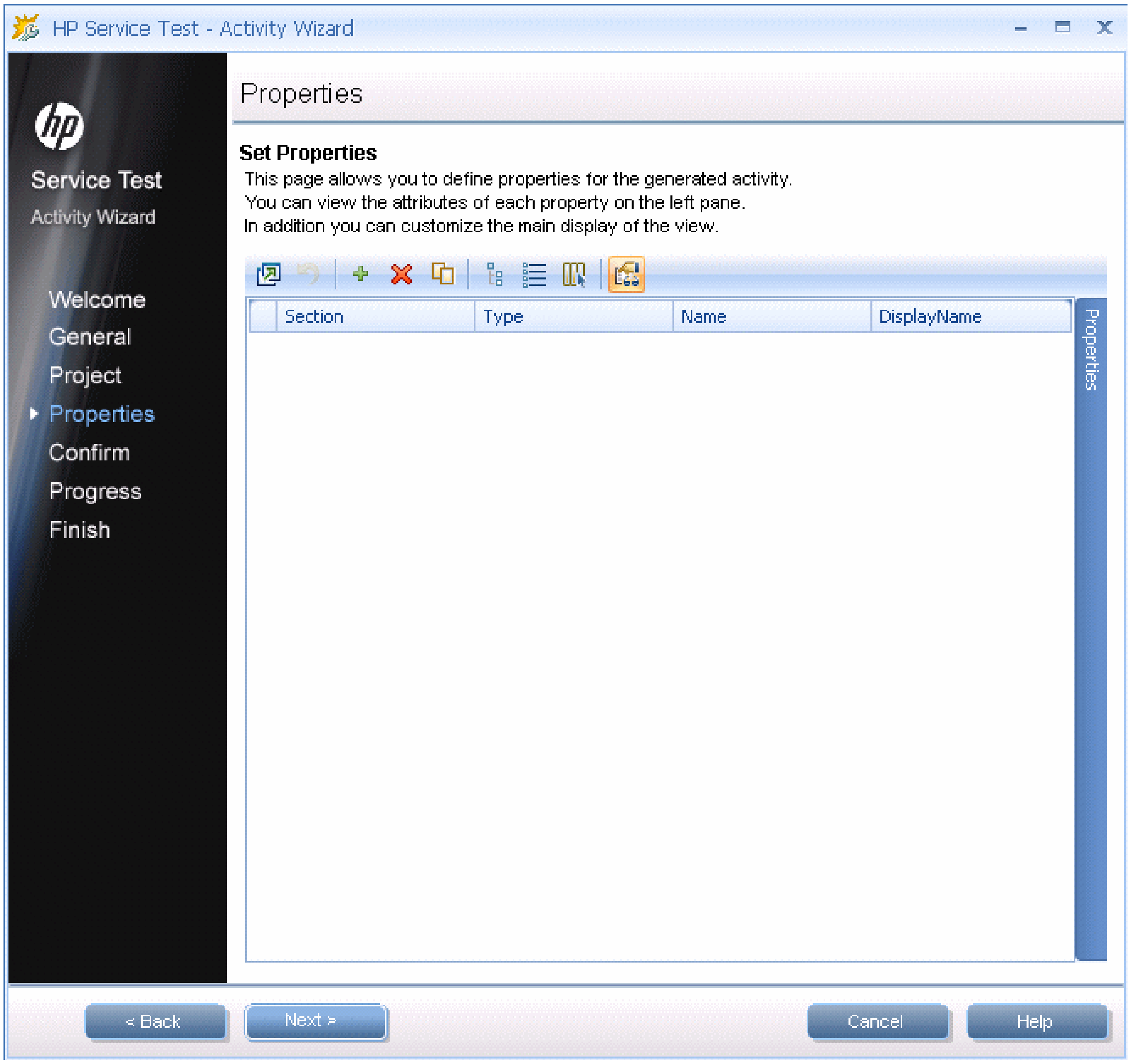
User interface elements are described below:

UI Elements	Description
Activity Class Name	A class name for the activity that will be generated.
Project Name	The name of the project.
Edit advanced properties	Enables the manual editing of additional project properties.
Project Namespace	The namespace to create for the project.
Project File Name	The name of the project file. By default, this is the **Project Name** with a .csproj extension.
Solution File Name	The name of the solution file. By default, this is the **Project Name** with an .sln extension.
Class FIle Name	The name of the class file. By default, this is the **Project Name** with a .cs extension.
Class Partial FIle Name	The name of the class file. By default, this is the **Project Name** with a .cs extension.
Assembly Name	The name of the assembly file. This name will be associated with the .dll extension.

Set Properties Page

Relevant for: API testing only

This wizard page lets you to define Input and Output properties for the activity and add new General properties.



Set Properties

This page allows you to define properties for the generated activity.
You can view the attributes of each property on the left pane.
In addition you can customize the main display of the view.

Important information	General information about this wizard is available here: "Activity Wizard" on page 2049.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

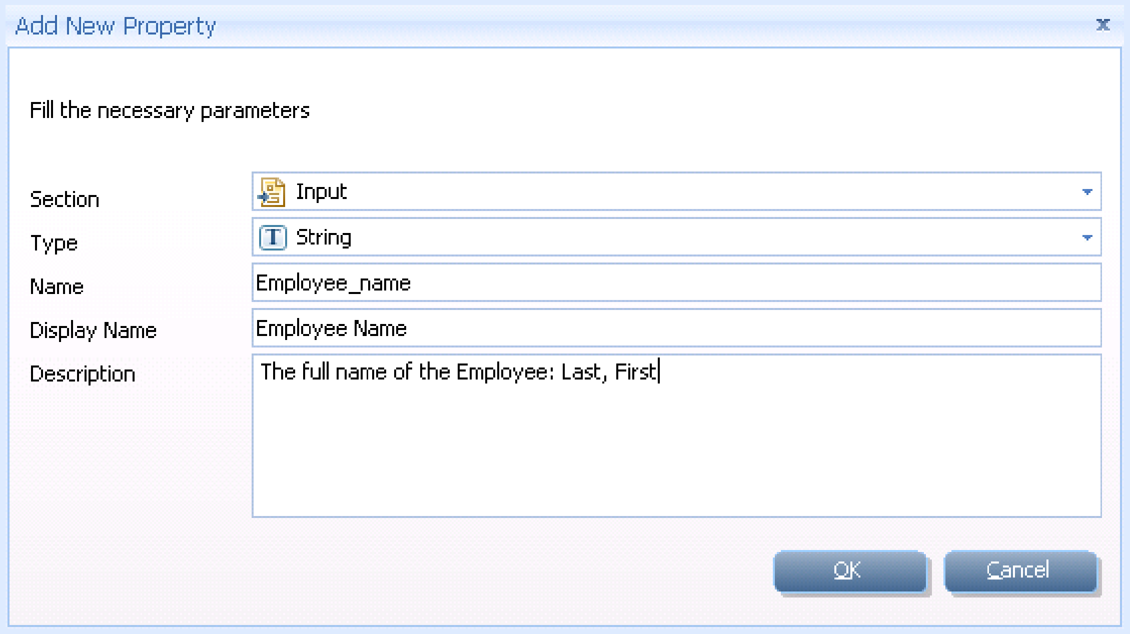
User interface elements are described below (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
⧉	**Open in Separate Window.** Opens a list of the properties in a separate window.
+	**Add Property.** Opens the Add New Property dialog box. For details, see "Add New Property" on the next page.
✕	**Remove Property.** Deletes the selected property from the activity.
⧉	**Clone Property.** Makes a copy of the selected property.
⋮	**Show Properties Tree.** Shows the properties in a tree hierarchy, by category—General, Input, and Output.
☰	**Show Properties List.** Hides the property tree hierarchy and shows all of the properties in a single grid.
▥	**Add/Remove Columns.** Opens the Add/Remove Columns dialog box enabling you to indicate which columns will be displayed in the Set Properties wizard page.
▤	**Show Properties Pane.** Opens the Properties pane in a dockable window. You can edit all of the property values— even ones that are not displayed in the wizard screen columns.
▤	**Hide Properties Pane.** Closes the Properties pane.
<property list>	A grid representation of the activities properties. **Tip:** - To show different or additional columns in the grid, click the **Add/Remove Columns** button and select the desired columns. - To change the order of the columns in the display, drag to column title to the desired location.

Add New Property

Relevant for: API testing only

The Add New Property dialog box lets you create new properties for the activity.



Fill the necessary parameters

Section	Input
Type	String
Name	Employee_name
Display Name	Employee Name
Description	The full name of the Employee: Last, First

To access	Do the following: 1. Open the Activity Wizard. For details, see "Activity Wizard" on page 2049. 2. Advance to the **Set Properties** page. 3. Click the **Add Property** button in the toolbar + .

User interface elements are described below:

Tree Elements	Description
Section	The section in which to add the property: **Input**, **Output** (Checkpoint), or **General**.
Type	The data type of the property, such as **String**, **Int**, and so forth.
Name	The name of the property as it will be called within the test and by event handlers.
Display Name	The name of the property as it will be displayed in the property list.
Description	An optional description of the property, that will appear in the hint area when you select the property in the Properties pane.

Confirm Page

Relevant for: API testing only

This wizard page shows you a summary of your settings before generating the activity.

Important information	General information about this wizard is available here: "Activity Wizard" on page 2049.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

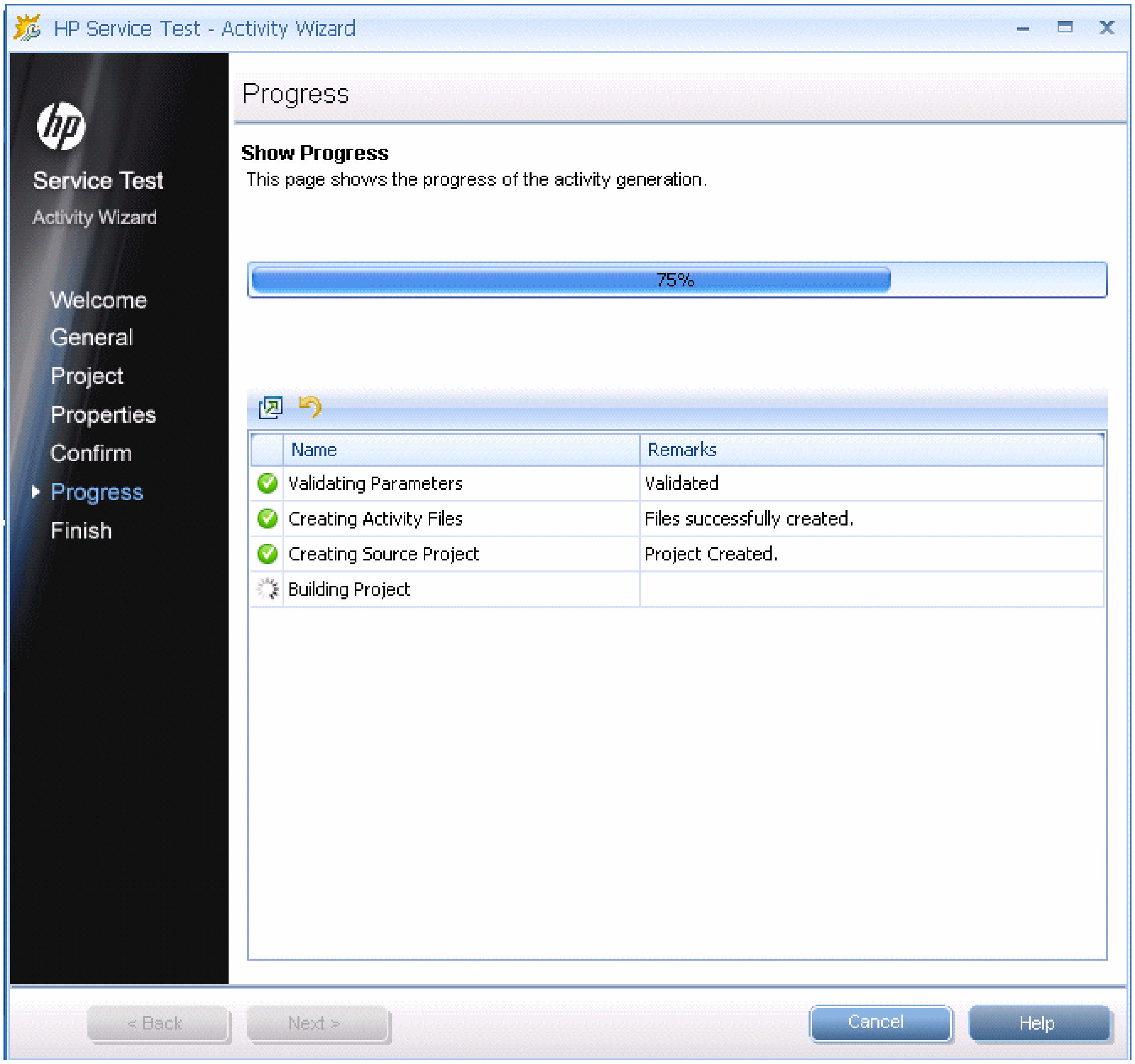
User interface elements are described below:

UI Elements	Description
<summary text>	A message indicating that the wizard is ready to begin the generation.

Progress Page

Relevant for: API testing only

This wizard page shows you the generation progress.



Show Progress

This page shows the progress of the activity generation.

Name	Remarks
✓ Validating Parameters	Validated
✓ Creating Activity Files	Files successfully created.
✓ Creating Source Project	Project Created.
Building Project	

Important information	General information about this wizard is available here: "Activity Wizard" on page 2049.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

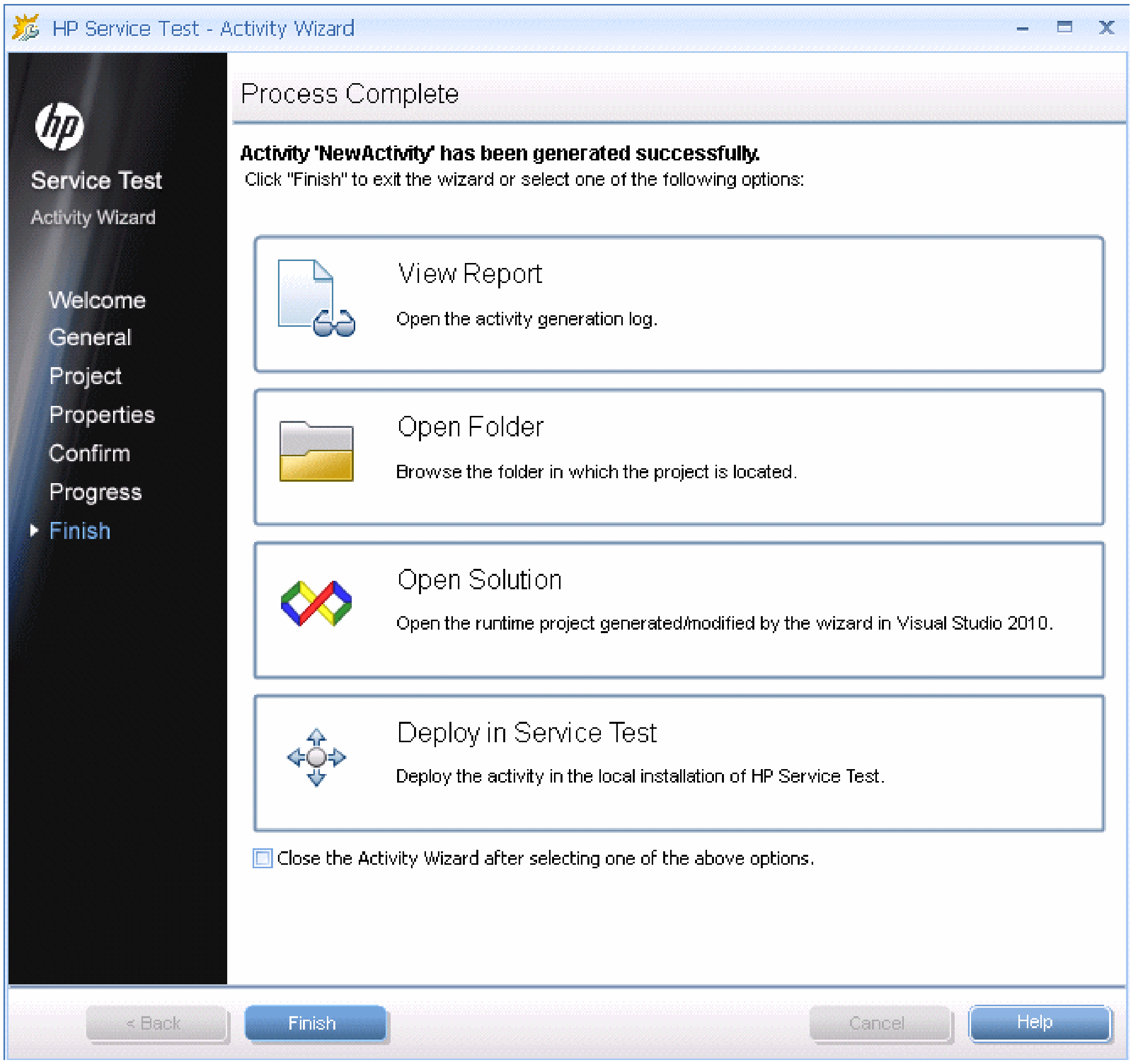
User interface elements are described below:

UI Elements	Description
<progress information>	A log indicating progress of the generation.

Finish Page

Relevant for: API testing only

This wizard page enables you to view the generation log, open the solution and its folder, and deploy the activity.



Activity 'NewActivity' has been generated successfully.
Click "Finish" to exit the wizard or select one of the following options:

View Report
Open the activity generation log.

Open Folder
Browse the folder in which the project is located.

Open Solution
Open the runtime project generated/modified by the wizard in Visual Studio 2010.

Deploy in Service Test
Deploy the activity in the local installation of HP Service Test.

□ Close the Activity Wizard after selecting one of the above options.

Important information	General information about this wizard is available here: "Activity Wizard" on page 2049.

Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

User interface elements are described below:

UI Elements	Description
View Report	Opens a log of the generation process. The log file is located in the %temp%\ ActivityWizard\Reports folder. Its title contains the generation date, WizardLog_#datetime#.log.
Open Folder	Opens the folder in which the activity files were generated. This folder is the Output directory you specified in the wizard. For example, you can open this folder to locate the java files to which you can add your custom code. The default subfolder is hp\st\ext\java.
Open Solution	Opens the activity's solution file in Visual Studio 2010. Use this option to edit and compile the source code.
Deploy in UFT	Deploys the activity in UFT by adding it to the **Toolbox** pane.
Close the Activity Wizard after selecting one of the above options	Closes the wizard screen automatically after selecting the **View Report**, **Open Folder**, **Open Solution**, or **Deploy in UFT** options.

Troubleshooting and Limitations - Extensibility (API Testing)

Relevant for: API testing only

This section describes troubleshooting and limitations for creating and using custom activities.

- You must have Visual Studio 2012 installed on the same machine as UFT to create custom activities using the Activity Wizard.

- The following error may occur if you place the signature file beneath a sub-folder of the **addin** folder.

  ```
  ServiceTest was unable to drag and drop the activity: Type 'http://hp.vtd.schemas/types/v1.0:
  GeneralPropertiesType' is not declared.
  ```

  Workaround: Modify the relative path of the Types schema. For example:

  ```
  .schemaLocation="../../dat/schemas/Types.xsd".
  ```

- If you modify the activity structure in the signature file, you will be unable to open tests using that activity. To modify an activity structure, create a new activity with the new structure, replace all of the test steps using the old activity, and then remove the old activity implementation.

- Custom Java activities created in versions of UFT prior to 12.00 will fail when you run your test.

  Workaround:

  a. Remove the ServiceTestCall.java interface from the project in which you created the custom Java activity..

  b. Change the project package name to something other than hp.st.ext.java.

  c. Add the ServiceTestCall.jar file to your classpath. This file is located in <UFT installation path>\Addins\ServiceTest\JavaCall\Java Interface\bin.

  d. Recompile your Java project.

Part 8: Business Process Testing in UFT

Chapter 64: Business Process Testing in UFT

Relevant for: business process tests and flows

This chapter includes:

Concepts

Business Process Testing Overview

Relevant for: business process testing

Business Process Testing provides you with a customizable, component-based testing framework that supports:

Component reuse and modularization

Component reuse and modularization keep costs low by speeding up test creation, maintenance, and execution.

Creation of tests for both simple and complex applications

An application under test can be a simple, HTML-based web application or a complex business process involving packaged applications and back-end services and databases.

Collaboration between various personas

The testing framework is flexible enough to meet the needs of various personas, such as manual testers, automation engineers, and subject matters experts.

Business Process Testing helps you document your components and tests, including screenshots illustrating how they should be used, and so on. This makes it easy for people with different roles and skill sets to share others assets.

Management of parts of a test

Managing parts of a test includes component documentation, test run results, version control, reporting, and history. Additionally, using ALM, you can generate documents containing information about the tests, flow, and components in a given project.

In Business Process Testing, you arrange components, including keyword GUI components, scripted GUI components, or manual components in a business process test or business process flow to test different scenarios. Business Process Testing enables you to arrange components to suit your needs.

> **Note:** Business Process Testing is available with ALM Edition and Quality Center Enterprise Edition. For more information about the HP Business Process Testing editions and their functionality, see the *HP Application Lifecycle Management User Guide*. To find out what edition of HP Business Process Testing you are using, ask your ALM site administrator.

Business Process Test and Flow Overview

Relevant for: business process testing

When working with Business Process Testing in UFT, you can use both **business process tests** and **business process flows** to organize your testing.

While tests and flows are fundamentally the same as they both contain a specified order of business components, there are differences:

- A business process test is a scenario comprising a sequence of business components or flows, designed to test a specific scenario in an application.
- A flow is a type of test that comprises a logical set of business components, in a fixed sequence, that performs a specific task. Flows share the same functionality as business process tests (for example, iterations, parameters, and results) in addition to functionality unique to flows (validation, debug mode). When designing flows, they can be considered as "compound components."

  In addition, flows cannot contain other flows.

  You can use a flow in multiple business process tests. When you modify a flow or any of its components, all business process tests containing that flow reflect that modification.

However, fundamentally, when you perform a run of your tests and flows, you must run them as part of a business process test.

Business Process Testing in UFT - Overview

Relevant for: business process tests and flows

In UFT, you can create and edit business process tests and flows with the native UFT Toolbox, Data, and Properties panes.

To work with BPT from within UFT, you must first connect to an ALM project with BPT support.

In UFT, you can create and edit keyword GUI components, scripted GUI components, and API components. For details about keyword and scripted GUI components, see "Business Components" on page 2080. API components are much like API tests, described in "API Testing Design" on page 1588. Some exceptions and limitations for working with API components are described in "Working with API components" on page 2079.

In addition, business components use application areas to store settings and resources that may be required by multiple components, such as shared object repositories and function libraries. Components are automatically linked to all of the resources and settings defined in the associated application area. For more details about application areas, see "Application Areas" on page 2095.

For a task on how to create and edit your business process tests in UFT, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

Business process tests and flows can also include manual testing components. For details about manual testing, as well as details about using BPT in ALM, see the *HP Business Process Testing User Guide* in the ALM documentation set.

Business Process Testing Methodologies

Relevant for: business process tests and flows

BPT is flexible and does not require any one particular model for incorporating business processes into your testing environment. The actual workflow in an organization may differ for different projects, or at different stages of the application development life cycle.

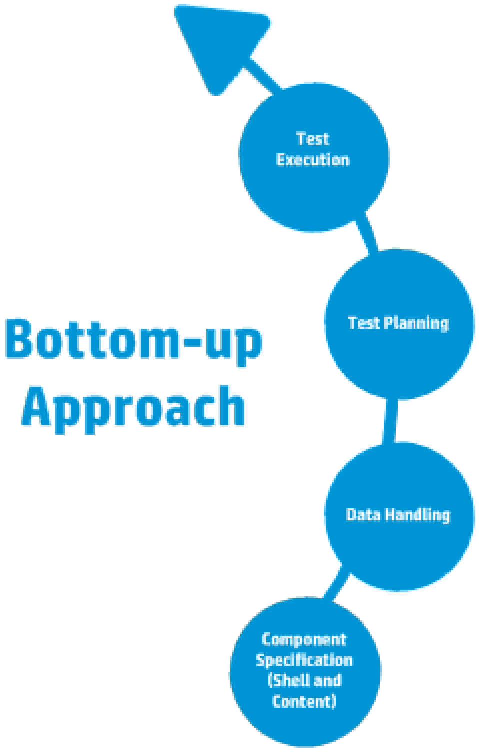
The chapters in this guide are structured according to the bottom-up methodology.

Bottom-up Methodology

Defining low-level components first and then designing business process tests based on the defined components is called a bottom-up methodology. This methodology is particularly useful:

- For regression testing
- When the business processes in the organization are clearly defined.
- When users are new to BPT.

The bottom-up methodology is based on the following design phases:



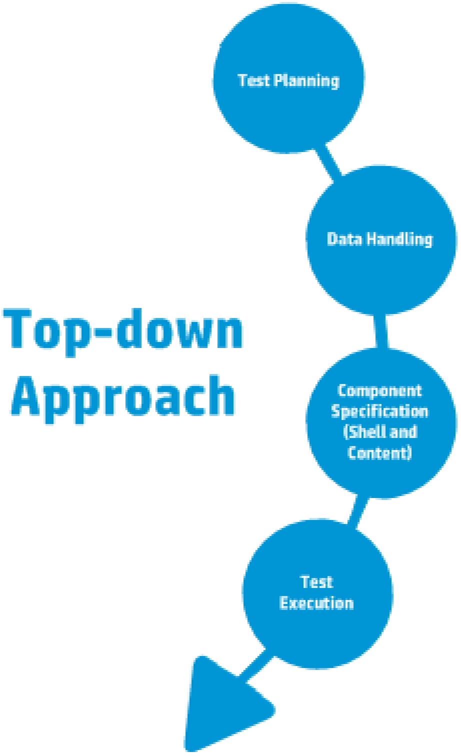
Phase	Description
Component Specification	• Develop a component tree with components. • Create the component shell by adding basic details. • Create component content by adding manual and/or automated implementations. Component content can contain: ▪ Manual implementation for manual components ▪ Automation, for automated components ▪ Both manual implementation and automation
Data Handling	Design the data that each business process test, flow, or component uses when run.
Test Planning	Build test plans and design business process tests and flows.
Test Execution	Create a subset of the business process tests in your project and run them.

Top-down Methodology

The top-down methodology advocates the creation of business process testing entities according to the following hierarchy:

- Business process tests, which contain flows and/or business components
- Flows, which contain business components
- Business components, which contain manual and/or automated steps

The top-down methodology is based on the following design phases:



Design Phase	Description
High-level design	Includes the high-level design, creation of a structure for business process tests, and determining the test configurations for testing different use-cases that will be needed. When designing at the high-level, facilitate automation: • By designing with modularity in mind. Design tests to use smaller, reusable components that automated tests can call multiple times. • By designing tests with reusable components, which makes maintaining tests easier.

Mid-level design	Includes the: • Creation of flows (sets of business components in a logical order that can be run). Flows are considered "compound components." • Creation of business components (reusable units that perform specific tasks in a business process). Only the shell of the component is created during this phase. • Specification of criteria for more granular test coverage (requirements) as necessary. • Linking to other testing documents. • Adding business components to business process tests and flows.
Low-level implementation	Includes the low-level implementation of business component content by: • Creating component steps (the content of the business component), including automated steps when necessary • Grouping components • Setting up iterations (for business process tests, flows, groups, and components) • Parameterizing

Agile Methodology

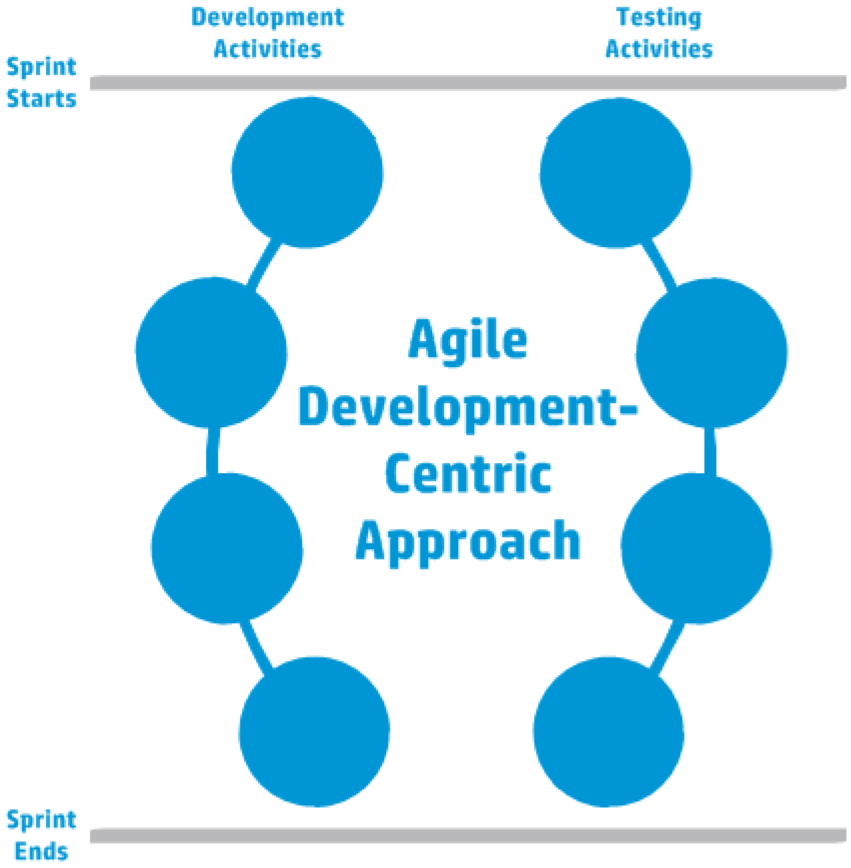
This approach is based on using BPT to provide testing in sprints, as developers code features for the application under test. Components and tests are created and updated in parallel with development.

> **Example**
>
> If the application under test is implemented in Java, components might be grouped by the classes that represent certain groups of UI elements, such as toolbar buttons. Each time a button is added to the toolbar, the component representing that class is updated.

This approach encourages:

- **Automation.** Because sprints are short, it is important to automate as much as possible.
- **Component reuse.** Component reuse can be designed in the same way that the developers implement modularly for reuse.

The following presents the Agile Development-centric approach.


Development Activities
Testing Activities
Sprint Starts
Agile Development-Centric Approach
Sprint Ends

Tasks

How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT

Relevant for: business process tests and flows

This task describes how to create, maintain, and run business process tests and flows in UFT.

This task includes the following steps:

- "Create a new business process test or flow or open an existing one" below
- "View and update test or flow properties" on the next page
- "Add content to business process tests and flows" on the next page
- "Use parameters in your test" on the next page
- "Iterate components and flows" on the next page
- "Group components and flows" on the next page
- "Debug and run your test" on page 2073

Create a new business process test or flow or open an existing one

"How to Create and Manage Documents" on page 136 describes how to create and manage business process tests and flows, as well as other UFT documents.

> **Note:** Make sure when opening UFT to edit a business process test that you select the same add-ins in the "Add-in Manager Dialog Box" (described on page 118)as the test objects contained in your component object repositories.

Additionally, you must save business process tests and flows on an ALM server, in a pre-defined test subject. For details about test subjects, see the *HP Business Process Testing User Guide*.

> **Note:** When you open a test or flow in UFT, it is automatically locked, and other users cannot modify it in UFT or ALM. When you close the test, it is automatically unlocked.
>
> If you open a test that is already locked by another user, the test opens as read-only. You can also open a test as read-only by selecting the **Open in read-only mode** option in the **Open test** dialog box. If changes are made to the test or flow by another user while you have it open (as read-only), click **Refresh** ⟳ in the BPT toolbar to update your test or flow with the latest data.

View and update test or flow properties

1. In the Solution Explorer, make sure that a test or flow is selected, and make sure that the Properties pane (described on page 438) is displayed.

   > **Note:** If you have a component or flow selected in the document pane, the Properties pane displays details about the selected component or flow instead of the test or flow selected in the Solution Explorer.

2. In the Properties pane General Properties tab (described on page 439), view basic information about the selected test or flow.

Add content to business process tests and flows

1. Open the Toolbox pane and navigate to the component or flow you want to add and double-click it or and drag and drop to add the component or flow to your flow or test.

2. In the document pane:

   - Order the items in your test or flow as needed by dragging and dropping the individual components or using the arrow buttons ⇧⇩ in the BPT toolbar.
   - To edit a specific component or flow, right-click the component name and select **Go to Component/Flow** . The component or flow opens as a new tab in the document pane, and the component or flow is added to the solution.
   - To remove a component or flow from your test or flow, select it and click **Delete from Test** ✕.

   For more details about the BPT toolbar, see "Toolbar Controls" on page 2075.

Use parameters in your test

For details, see "How to Use Data in a Business Process Test" on page 2124

Default values for component or flow parameters are used during the run session, if no other value is supplied.

Iterate components and flows

By default, each component or flow you add to your test has a single iteration. For details on defining iterations and parameter values for the iterations, see "Add iterations for a component or flow" on page 2126 and "Set data values for each iteration" on page 2126.

Group components and flows

In some cases, it may be helpful to manage several business components or flows together as a group.

In the document pane, select the components or flows you want to group, and click **Group** . To ungroup, select the group and click **Ungroup** .

> **Note:** When iterating groups, all items to be included in the group must have the same number of iterations, or an error message is displayed when you group the items.

Debug and run your test

- **To debug a test or component**, insert breakpoints in specific components or flows in your test, and then run your test. The run session stops at each breakpoint. For details, see "How to Use Breakpoints " on page 687.

  > **Note:** If you run a BPT test from the ALM Test Plan module or from UFT, UFT stops the test at all breakpoints in both **Debug** and **Normal** modes. However, before running the BPT test, you **must open** the business components with the breakpoints and add them to the solution in UFT.

- **To run your test**, do one of the following:

  - Click the **Run** button or select **Run > Run** to select the test or flow to run from any of the open documents in Run dialog box.

  - Select **Run > Run Now** to run the test or flow in focus in the document pane.

  > **Note:**
  >
  > - Before running a test, you must enable the **Allow other HP products to run tests and components** option in the Test Runs pane (**Tools > Options > GUI Testing** tab > **Test Runs** node)
  >
  > - For improved performance when running business process tests or flows, UFT creates and runs a hosting test, named **Test Runtime**. The **Test Runtime** test is recreated each time the test or flow runs, and is not saved with the run.

During the test run, click **Pause** or **Stop** in the toolbar to pause or stop the run session.

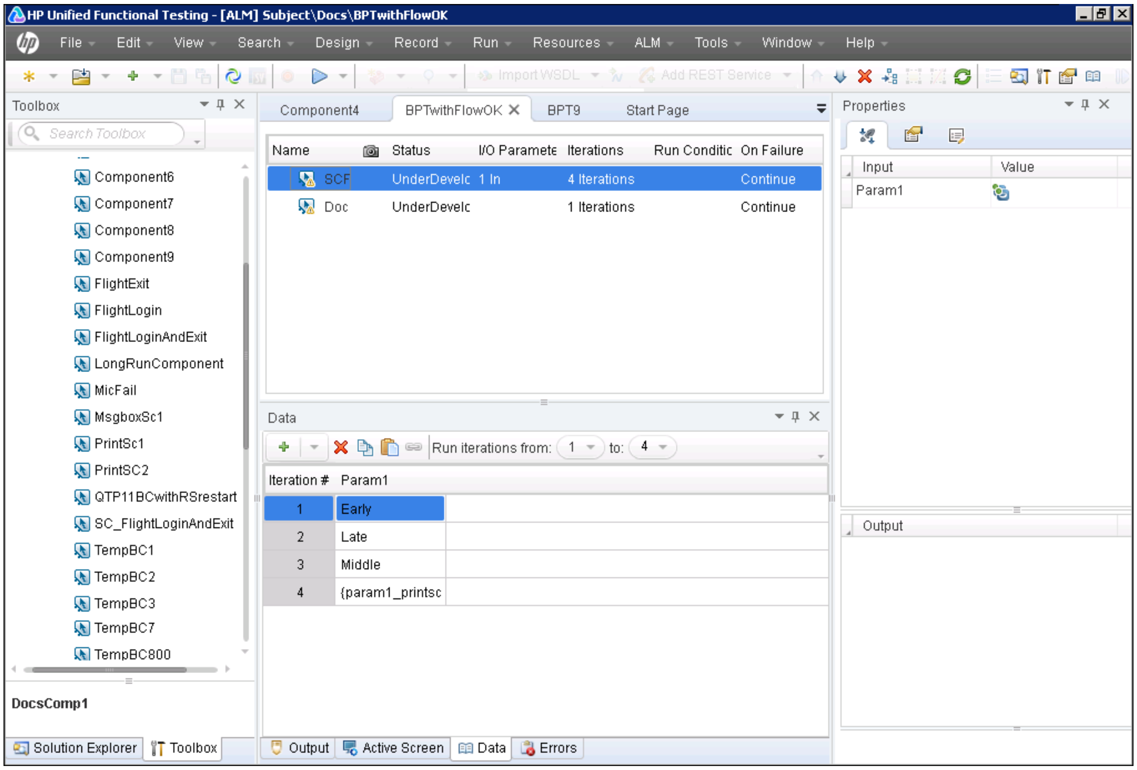
By default, when the run session ends, the Run Results Viewer opens and the test results are displayed. For details about analyzing the run results, see the *HP Run Results Viewer User Guide*.

Reference

Business Process Testing in UFT User Interface

Relevant for: business process tests and flows

Business process tests and flows are displayed in the document pane in a grid. Selecting a specific component, flow, or group in the document pane causes information to change in the Properties and Data panes. For general information about the document pane, see "Document Pane Overview" on page 303.



Name		Status	I/O Paramete	Iterations	Run Conditic	On Failure
SCF		UnderDevelc	1 In	4 Iterations		Continue
Doc		UnderDevelc		1 Iterations		Continue

Iteration #	Param1
1	Early
2	Late
3	Middle
4	{param1_printsc

Input	Value
Param1	

To access	"Create a new business process test or flow or open an existing one" on page 2071. The business process test or flow opens as a tab in the document pane.
Important information	• "Business Process Testing Overview" on page 2064 • "Business Process Test and Flow Overview" on page 2064 • "Business Process Testing in UFT - Overview" on page 2065
Relevant tasks	"How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071

User interface elements are described below:

- "User Interface Elements (Grid Columns)" below
- "Toolbar Controls" below
- "Context menu items" on the next page

User Interface Elements (Grid Columns)

UI Element	Description
Name	The name of the component or flow.
📷	**Snapshot.** Indicates whether a snapshot exists for this component.
Status	The ALM workflow status for the components and flows in the test.
I/O Parameters	The number of parameters defined for the selected component or flow, as defined in the "Parameters Tab (Properties Pane - BPT)" on page 444.
Iterations	The number of iterations defined in the pane for the selected component or flow.
Run Conditions	The attributes that must match for the component to run, as defined in the "Properties Tab (Properties Pane - BPT)" on page 447. **Note:** You can modify these attributes only when editing a flow. This value is blank for components contained in a test.
On Failure	The action for UFT to take if a specific business component in the flow fails, as defined in the "Properties Tab (Properties Pane - BPT)" on page 447. Valid values include: • **Exit.** The business process test run ends if the selected business component fails. • **Continue.** (Default) The business process test will run the next business component or flow if the selected component fails.

Toolbar Controls

The following toolbar buttons are included in the BPT toolbar above the document pane.

Button	Description
⇧ ⇩	**Move up/down.** Enables you to change the order of entities in the business process test or flow by moving a selected component, group, or flow up or down.

Button	Description
✕	**Delete from Test.** Removes the selected business component, group, or flow from the business process test or flow. **Caution:** If you are deleting the last component in a group, the entire group is deleted.
	Go to Component/Flow. Opens the selected component or flow in the document pane. The component or flow is also added to the solution and listed in the Solution Explorer.
⬚	**Group.** Creates a group that includes the selected business components and/or flows. The components and flows must be contiguous. A component or flow can belong to one group only. A group node is created above the grouped items and is identified by the group icon. By default, the group is named Group, followed by a unique number. **Tip:** • You can add other business components or flows to an existing group by dragging and dropping a component or flow from the Toolbox pane to the relevant position in the group. • You can change the order of the members of the group by dragging and dropping.
	Ungroup. Ungroups components and/or flows. To completely remove a group, including its members, select the group and click **Delete from Test** ✕. To remove a business component or flow from a group, select the component or flow, drag it up or down out of the group, and drop it at the required location.
🔄	**Refresh.** Refreshes a currently open read-only test with any updated information.

Context menu items

The following table describes the context menu items available from the document pane, when editing a business process test or flow.

Button	Description
↑ ↓	**Move up/down.** Enables you to change the order of entities in the business process test or flow by moving a selected component, group, or flow up or down.
✕	**Delete from Test.** Removes the selected business component, group, or flow from the business process test or flow.
	Go to Component/Flow. Opens the selected component or flow in the document pane. The component or flow is also added to the solution and listed in the Solution Explorer.
	Group. Creates a group that includes the selected business components and/or flows.
	Ungroup. Ungroups components and/or flows.

Comparing BPT in UFT to BPT in ALM

Relevant for: business process tests and flows

Business process tests and flows are displayed in the UFT document pane, in a display similar to the ALM Test Script tab. The following table describes where to find the BPT functionality you are familiar with from working in ALM, when you are working in UFT.

For details about using BPT in ALM, see the *HP Business Process Testing User Guide*.

Function	ALM	UFT
Create or open test or flow	Test Plan > New or Details button	"Open/New <Document>/<Resource> Dialog Box" on page 156
Check in and check out tests and flows	Version Control user interface	"Version Management Commands" on page 802
View test properties	Test Plan > Test Details dialog box > Details tab	"General Properties Tab (Properties Pane - BPT)" on page 439
View, add, or modify test parameters	Test Plan > Parameters tab	"Test Parameters Tab (Properties Pane - BPT)" on page 440
Add component or flow to test or flow	Test Plan > Test Script tab > Select Components and Flows pane	"Toolbox Pane User Interface (BPT in UFT)" on page 518

Function	ALM	UFT
Reorder, group, or open components or flows in tests View run conditions or failure settings for components or flows	Test Plan > Test Script tab	"Business Process Testing in UFT User Interface" on page 2074
Edit run conditions for components in flows	Test Plan > Test Script tab > Run Conditions button	"Properties Tab (Properties Pane - BPT)" on page 447
Edit failure settings for components in flows	Test Plan > Test Script tab > On Failure link in grid	"Properties Tab (Properties Pane - BPT)" on page 447
Add or modify component parameters	Business Components > Parameters tab	"Parameters Tab (Properties Pane - BPT)" on page 444
Link parameters	Test Plan > I/O Parameters link in grid > Link I/O column > Select Output Parameter dialog box	"Select Link Source Dialog Box (BPT)" on page 2128
Promote parameters	Test Plan > Test Script tab > Select Components and Flows pane > QuickAdd button > Add While Setting Promote Options	"Parameters Tab (Properties Pane - BPT)" on page 444 "Data Pane User Interface (BPT in UFT)" on page 265 "BPT Testing Tab (Options Dialog Box)" on page 570
View and modify BPT comments for a specific component instance	Test Plan > Test Script tab > Comments tab	"Component Details Tab (Properties Pane - BPT)" on page 451
Add and modify iterations and iteration data for components, flows, and groups	Test Plan > Test Script tab > Iterations link in grid	"Data Pane User Interface (BPT in UFT)" on page 265
Run or debug test	Test Plan > Test Script tab > Run or Debug Test button	"Run Dialog Box" on page 651

Troubleshooting and Limitations - Business Process Testing in UFT

Relevant for: business process tests and flows

This section includes:

- "General functionality" below
- "Working with API components" below

General functionality

To create BPT tests and flows in UFT, you must work with ALM 11.52 or later.

Working with API components

You can create and manage API components in much the same way as you would API tests, as described in "API Testing Design" on page 1588. The following exceptions apply when working with API components:

- You cannot use Load Test activities in a component. If you save a load-enabled test as a business component, it will no longer be load-enabled.
- You cannot use Automated Testing Tool activities in a component.
- Encoded password type properties are not supported. If the component has a property of type password (or encrypted in ALM 11.00 and later), the value will be treated as an ordinary string, without encoding or decoding.
- Multiple User Variable profiles are not supported. Remove all but one of the profiles.

Note: Other general differences between test and components also apply to API components, such as "Troubleshooting - Naming Conventions" on page 2269

Chapter 65: Business Components

Relevant for: GUI components only

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

For a list of the supported versions of ALM or Quality Center, see the *HP Unified Functional Testing Product Availability Matrix*, available from the UFT Help or the root folder of the Unified Functional Testing DVD. The most up-to-date product availability matrix is available from the HP Software Product Manuals site, at http://h20230.www2.hp.com/selfsolve/manuals (requires an HP Passport).

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

This chapter includes:

Concepts

Business Components - Overview

Relevant for: business process tests and flows

You can create, define, modify, and manage business components in UFT or ALM. Business components provide the basis for BPT and are incorporated into business process tests and flows.

A business component is a reusable unit that:

- Performs a specific task in a business process
- Describes the condition or state of the application before and after that task

You can use a component in multiple business process tests and flows. When you modify a component or its content, all business process tests or flows containing that component reflect that modification. You can also use run conditions to enable components to run selectively, based on earlier stages within the test or flow.

Business Component Examples

Business Component Name	Task	Application State Before	Application State After
Login	Banker logs into banking application	<none>	The application is launched and the main home page is displayed.
SearchLoan	Banker searches for an existing loan	The banker is logged in and the main home page is displayed.	The application displays the main loan details page, or a page indicating that the loan was not found.

Business components are comprised of:

- **A Shell** (general information such as component name, status).
- **Content** (steps or scripts). Low-level, detailed information, such as the component's manual steps and/or automation. The contents provide detailed instructions for performing business process tasks in the application. Component content can be manual, automated, or both—depending on whether you create a manual implementation and/or automation for the component.
    - For details on automated components, see "Keyword GUI Components Overview" on page 2084 and "Scripted GUI Components Overview" on page 2085.

Example of Content

Step	Description	Expected Result
1	Open the application.	The application is launched and the login page is displayed.
2	Enter a user name.	The cursor advances to the password field.
3	Enter a password.	The password is displayed as asterisks.
4	Click **Submit** on the Web page.	The application's main page is displayed.

API components are much like API tests, described in "API Testing Design" on page 1588. Some exceptions and limitations for working with API components are described in "Working with API components" on page 2079.

- For details about manual testing in BPT, see the *HP Business Process Testing User Guide* in the ALM documentation set.

This section also includes:

Business Component Categories

Relevant for: business process tests and flows

Because BPT is a component-based testing framework, components are largely responsible for driving the actual application testing. This component framework encourages component design and reuse, so the method you use to categorize your components has a large impact on the ability of your framework to manage your testing abilities successfully.

You can use one or more of the following methods to categorize components .

- "Logical Components" on the next page
- "Application Object Components" on the next page
- "Generic Components" on the next page

Logical Components

A logical component represents the use of a part of the screen with one or more controls, or a set of API calls which combine to perform some application logic. This category is based on a specific context in the application under text.

Examples

- A **Login** component represents the login process, based on a login window that allows you to enter a user name and password, and then click a **Login** button.
- A **Search** component represents search for an entity in the application under test. You can enter a string for which to search, indicate capitalization and/or whole word options, and click a **Search** button.

Application Object Components

An application object component might represent an object on the screen or a call to a single API.

This category is usually independent of the context within the application under test, and can be used in many situations. You decide the level of granularity that most encourages reuse.

Examples

- A **Button** component represents the button object.
- A **Grid** component represents a grid object in a pane or window.
- A **Pane** component represents a pane in a window or screen.
- An **Interrogate** component represents the interrogation of the application under test's backend database.

Generic Components

A generic component performs actions outside of the context of the application under test. It can be reused in tests of different applications.

Example

- A **Launch** component represents the launching of a browser.

Flows can be thought of as complex components or small business component tests. Flows comprise a set of components in a fixed sequence to perform a specific task. A flow can be part of a test just like any other component, but when the flow runs, BPT executes the components that the flow contains.

Keyword GUI Components Overview

Relevant for: Keyword GUI components only

You can use the Keyword View to create, view, and modify a keyword component in UFT.

In the Keyword View, keyword components are divided into steps in a modular, keyword-driven, table format. Each step is a row that comprises individual parts that you can easily modify. You create and modify steps selecting items and operations and entering additional information, as required.

Each step in a keyword component is automatically documented as you complete it. This enables you to view a description of the step in understandable sentences. In addition, if you added a function library to the application area associated with the keyword component, when you define a step by selecting a user-defined operation (function), the documentation that you added in the function library is displayed for the step. For details, see "How to Create and Work with a User-Defined Function" on page 1048.

> **Note:** ALM users can access components from the ALM Business Components module. For details, see the *HP Business Process Testing User Guide*.

Keyword GUI Component Resources

- Each keyword component is based on a specific application area, which is stored in the ALM project in which you intend to save the component.
- Each application area specifies the settings and resources for the keyword component, including the location of shared object repositories, function libraries, recovery scenarios, and other information.
- There may be one or more application areas from which to choose. You select the application area that is best suited for your keyword component. For details, see "Application Areas" on page 2095.

This section also includes:

- "Tips and Guidelines for Selecting Application Areas for Your Component" on the next page

Tips and Guidelines for Selecting Application Areas for Your Component

Relevant for: Keyword GUI components only

- When you create a GUI component in UFT, you must select the application area to which you want to associate the component. There may be one or more application areas available from which to choose. You should select the one that is best suited for the component.

- If changes are made to your application or to the resource files and settings associated with the application area, the application area may become unsuitable, and you may need to associate a different application area with a specific component. For example, the object repository could have been modified or removed from the application area.

- As your application develops, it may include additional or different objects that are not contained in the currently associated object repository. This could cause the component or business process test to run incorrectly or to fail.

- If another application area contains the required resource files and settings, you should associate that application area with the component.

- Each time you open a component, UFT verifies that the resources specified for the component are available. If a component or application area has resources that cannot be found, such as a missing shared object repository, UFT indicates this in the Errors pane. For details, see "Errors Pane" on page 364.

Scripted GUI Components Overview

Relevant for: Scripted GUI components only

Scripted components are maintainable, reusable scripts that perform a specific task.

Using scripted components enables you to utilize the full power of both the Keyword View and the Editor, as well as other UFT tools and options to create, view, modify, and debug scripted components. For example, you can:

- Use the Step Generator to guide you through the process of adding methods and functions to your scripted component.

- Use the Editor to enhance the scripted component flow by manually entering standard VBScript statements and other programming statements using test objects and methods.

- Incorporate user-defined functions in your scripted component steps, parameterize selected items, and add checkpoints and output values.

This section also includes:

- "Use Scripted GUI Components for Complex Functionality" below
- "Similarities Between Scripted GUI Components and Tests" below
- "Similarities Between Scripted and Keyword GUI Components" on the next page
- "Differences Between Scripted and Keyword GUI Components" on the next page
- "Converting Keyword GUI Components to Scripted GUI Components" on the next page

Use Scripted GUI Components for Complex Functionality

You can create scripted components that contain more complex functionality, such as loops or conditional statements. You can then include these scripted components in business process tests to check that the application behaves as expected.

> **Note:** For details about viewing components in tests and flows in UFT, see "Business Process Testing in UFT User Interface" on page 2074.
>
> In the Component Steps tab of the ALM Business Components module, ALM users can view and work only with the manual steps defined for a scripted component (if any). They cannot view or modify the automated steps unless they open the scripted component in UFT by clicking the **Launch** button in the Automation tab (provided that UFT is installed on the ALM client). For more details, see the *HP Business Process Testing User Guide*.

Similarities Between Scripted GUI Components and Tests

Scripted components contained much of the same functionality as actions and tests. For example, you can:

- Work with programmatic statements in the Editor (see "Programming in GUI Testing Documents in the Editor" on page 994).
- Use standard and bitmap checkpoints and standard output values.
- View the hierarchical Keyword View display.
- Create and work with virtual objects.
- Use the Active Screen to view a snapshot of your application as it appeared when you performed a certain step during a recording session, and to parameterize object values and insert checkpoints, methods, and output values for any object in the page, even if your application is not available or you do not have a step in your test corresponding to the selected object.
- Use random and environment parameters.
- Set applications to open at the start of a record or run session.
- Use the analog and low-level recording modes.

Similarities Between Scripted and Keyword GUI Components

Both scripted and keyword components are:

- Associated with a specific application area. Therefore, all of the resources must be stored in the ALM project and not in the file system.
- Standalone modular units that can be incorporated into a business process test.
- Linear within a business process test (not hierarchical).
- Not nested, meaning they cannot call another component.

Differences Between Scripted and Keyword GUI Components

Scripted components cannot be modified in ALM.

Converting Keyword GUI Components to Scripted GUI Components

You can convert keyword components to scripted components, when required. However, because scripted components can be modified only in UFT (and not in ALM), the automated steps cannot be viewed in ALM.

The conversion process is not reversible, meaning you cannot convert the scripted component back to a keyword component.

Tasks

How to Create and Manage GUI Business Components

Relevant for: GUI components only

This task describes the different operations you can perform to manage business components, and contains general considerations and guidelines.

This task includes:

- "Prerequisites" below
- "Guidelines for users of earlier QuickTest versions" below
- "Create a new business component" on the next page
- "Open a business component" on the next page
- "Save a business component" on page 2090
- "Associate a different application area with your component" on page 2090
- "Delete a component" on page 2090

Prerequisites

- You must connect UFT to an ALM project, which is where components and application area resources and settings are stored. Connecting your ALM project enables UFT to create or open the component. This also enables the component to access all of the resources defined in the application area on which the component is based. For details, see "ALM Connection Dialog Box" on page 737.

- Make sure you have the required ALM permissions before working with components and application areas. For details on setting user group permissions in the Business Components module, see the *HP Business Process Testing User Guide* and the *HP Application Lifecycle Management Administrator Guide*.

- Components that are currently open in ALM or another UFT session are locked and can be opened only in read-only format. To work with these components, they must be closed everywhere else. Additionally, if the resources used by a component are locked, these resource files open in read-only mode. For details on working with locked resources , see "Opening and Saving Tests with Locked Resources" on page 134.

Guidelines for users of earlier QuickTest versions

- To modify a component last modified using a version of QuickTest earlier than 9.5, you must upgrade the component to the QuickTest 11.00 format using the QuickTest Asset Upgrade Tool for ALM (found on your QuickTest 11.00 installation DVD).

After upgrading the version of the component, open the upgraded component in UFT. Before it is upgraded, though, you can view it in read-only format, and you can run it. (To work with a component last modified using QuickTest version 8.x, it must first be upgraded to QuickTest version 9.x.) All components last modified in versions of QuickTest later than 9.5 can be opened in UFT without conversion.

- After you save a converted component, it cannot be used with earlier versions of QuickTest.

Create a new business component

1. Select **File > New > Business Component**.

   In the "New <Document> Dialog Box" (described on page 152), select a folder in the Business Components module in ALM in which to store your component and give your component a name.

2. In the **Application Area** field of the "New <Document> Dialog Box", click the **Browse** button to select a suitable application area from within the **ALM Test Resources** module. After choosing your application area, click **OK.**

3. A new business components opens in the Keyword View (for keyword components) or in the Editor (for scripted components).

   Although the component does not yet contain content, it does contain all of the required settings and resources that were defined in the application area on which it is based. You can view these settings in read-only format by choosing **File > Settings**. If you later need to change these settings, you can do so in the associated application area.

4. Optional - You can now add steps and comments to your business components.

   For details relevant for **keyword components**, see "How to Add a Standard Step to Your Test or Component" on page 925 and "Comments in the Keyword View" on page 922.

   For details, relevant for **scripted components**, see "How to Enhance Your Actions, Scripted Components, and Function Libraries Using the Windows API" on page 1011 and "Programming in GUI Testing Documents in the Editor - Overview " on page 995.

Open a business component

Do one of the following:

- Select the component in the Open Business Component dialog box, which you open by selecting **File > Open > Business Component**. For details, see "Open/New <Document>/<Resource> Dialog Box" on page 156.

- Select the component from the list of **Recent** files in the **File** menu.

Save a business component

- To save a modified component, select **File > Save**. The component is saved with the changes.
- To save a new component or to save an existing component with a new name, select **File > Save As**.

  The business component is saved to the component tree in the ALM project (Business Components module).

  > **Note:** (**for scripted GUI components:** When working with scripted components, the data sheet name in the Data pane is identical to the scripted component name. Therefore, if you save a scripted component with a new name using **File > Save As**, the data sheet is automatically renamed. If you have a step that references the data sheet by name, the step will fail during the run session because it references the former data sheet name. If you save a scripted component with a new name, you must find any references to the former data sheet name in the Editor and replace them with the new data sheet name.

Associate a different application area with your component

Do one of the following:

- Select **File > Change Application Area**.
- Right-click on a component node and select **Change Application Area**.

In the "Change Application Area Dialog Box" (described on page 2092), you select a different application area and click **OK** to change the application area associated with a component.

Delete a component

You delete a component in ALM, regardless of whether it was created in UFT or in ALM. For details, see the *HP Business Process Testing User Guide*.

How to Convert a Keyword GUI Component to a Scripted GUI Component

Relevant for: GUI components only

This task describes how to convert a single keyword component to a scripted component.

> **Caution:** This replaces the existing keyword component with a scripted component, and cannot be undone.

This task includes the following steps:

- "Prerequisites" below
- "Convert the keyword GUI component to a scripted GUI component" below
- "Results" below

1. **Prerequisites**
   - You must connect UFT to an ALM project. For details, see "ALM Connection Dialog Box" on page 737.
   - The keyword component must be editable (meaning it must not read-only mode or locked by another user). For example, if your are working with version control, you must check out the component.
2. **Convert the keyword GUI component to a scripted GUI component**
   a. Open the keyword component you want to convert.

   b. Select **File > Convert to Scripted Component**. When prompted, click **OK** to proceed with the conversion.
3. **Results**

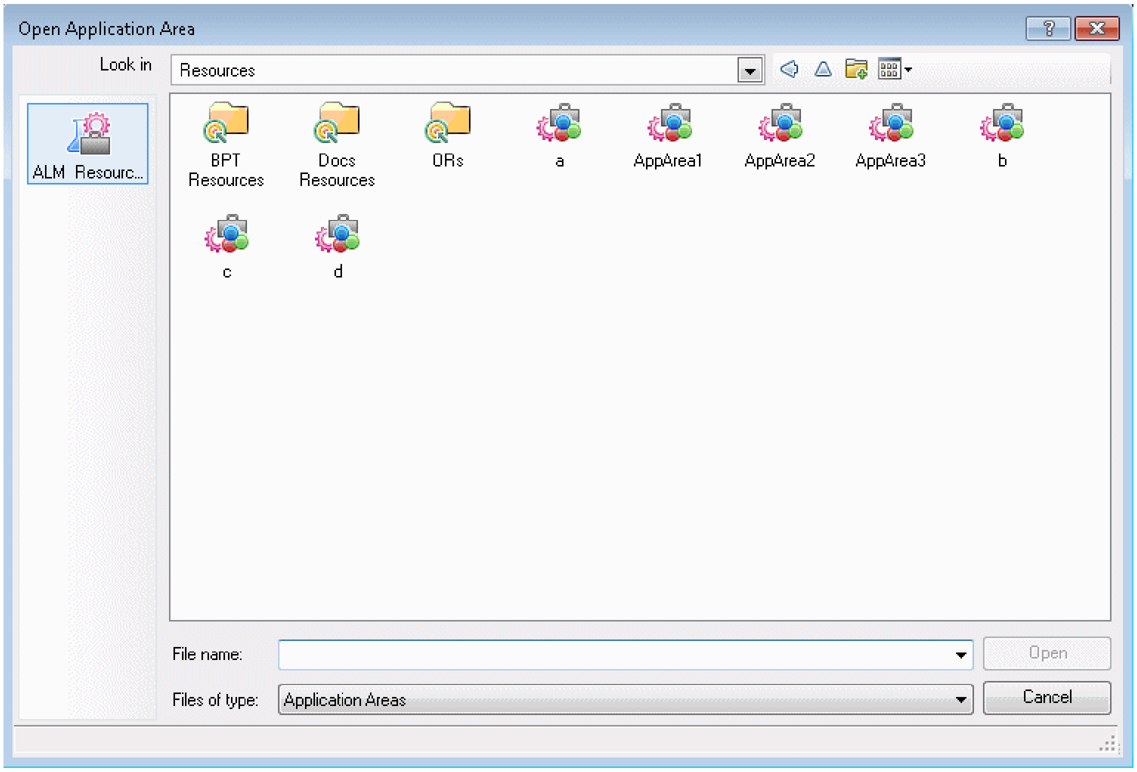
   After the conversion is complete, the scripted component opens automatically.

Reference

Change Application Area Dialog Box

Relevant for: Keyword GUI components only

This dialog box enables you to associate an existing component with another application area.



To access	1. Open a business component or select the component node in the Solution Explorer. 2. Do one of the following: ▪ Select **File > Change Application Area**. ▪ Right-click the node of a component and select **Change Application Area**.
Important information	This dialog opens to the ALM Resources module in your ALM project. All available application areas contained in this module are displayed.
Relevant tasks	"How to Create and Manage GUI Business Components" on page 2088
See also	"Keyword GUI Components Overview" on page 2084

User interface elements are described in the "Open/New <Document>/<Resource> Dialog Box" on page 156.

Troubleshooting and Limitations - Business Components

Relevant for: GUI components only

This section describes troubleshooting and limitations for business components.

- For troubleshooting and limitations for learning objects, and recording and running steps, see "Learning objects, running steps, and recording steps (GUI testing only)" on page 661.

- **Start Menu / Quick Launch Panel.** On Windows Vista, Windows 7, Windows Server 2008, Windows Server 2008 R2, Window 8 ,or Windows Server 2012, if you begin a recording session after installing UFT (or upgrading from QuickTest) without restarting your computer, UFT cannot record operations on the Windows **Start** menu or **Quick Launch Panel.**

  Workaround: Restart your computer and start a new recording session.

- **Start menu.** UFT does not record launching Windows Help from the **Start Menu**.

- **Start Menu.** UFT does not record the selection of **Start** menu items that are customized as menus, such as My Computer, Control Panel, or Recent Documents.

  Workaround: Customize the **Start** menu items as links so that UFT can record operations on them or record the activation of these items another way (and not through the **Start** menu).

- Renaming a UFT component from ALM may cause the test or component to behave unexpectedly.

  Workaround: To rename a test or component, open it in UFT and rename it using the **Save As** option. If the test or component has already been renamed from ALM, use the rename option again to restore the old name, and then use the **Save As** option in UFT. Renaming a UFT test parameter from UFT will cause any run-time parameter values already set for this parameter in ALM to be lost.

- **Multilingual Support.** Names and paths of components, application areas, and resources (for example, function libraries, object repositories, and recovery scenarios) are not Unicode compliant and therefore should be specified either in English or in the language of the operating system.

- Business component scripts that were created in QTP 11.00, and contain the **ActionName** environment variable, return different values for the **ActionName** variable value when run in UFT.

  For example, an **ActionName** variable that returned the BC1 value as the component name in QTP 11 returns the BC1[BC1] value in UFT.

  Workaround: Update your scripts as needed to use the new environment variable value.

Chapter 66: Application Areas

Relevant for: GUI components only

> **Note:** Unless otherwise specified, references to **Application Lifecycle Management** or **ALM** in this guide apply to all currently supported versions of ALM and Quality Center. Note that some features and options may not be supported in the specific edition of ALM or Quality Center that you are using.
>
> For a list of the supported versions of ALM or Quality Center, see the *HP Unified Functional Testing Product Availability Matrix*, available from the UFT Help or the root folder of the Unified Functional Testing DVD. The most up-to-date product availability matrix is available from the HP Software Product Manuals site, at http://h20230.www2.hp.com/selfsolve/manuals (requires an HP Passport).
>
> For details on ALM or Quality Center editions, see the *HP Application Lifecycle Management User Guide* or the *HP Quality Center User Guide*.

This chapter includes:

Concepts

Application Areas - Overview

Relevant for: GUI components only

When you create a set of components to test a particular area of your application, you generally need to work with many of the same test objects, keywords, testing preferences, and other testing resources, such as function libraries and recovery scenarios. You define these files and settings in an application area, which provides a single point of maintenance for all elements associated with the testing of a specific part of your application.

An **application area** is a collection of settings and resources that are required to create the content of a business component. Resources may include shared object repositories containing the test objects in the application being tested, function libraries containing user-defined operations that can be performed on that application, and so forth. Components are automatically linked to all of the resources and settings defined in the associated application area.

For example, to be able to select test objects when implementing steps for a component, the test objects must be stored in a shared object repository associated with the component's application area. Associating function libraries with your application area enables you to access the functions defined in these function libraries as keywords.

You can create as many application areas as needed. For example, you may decide to create an application area for each Web page, module, window, or dialog box in your application. Alternatively, for a small application, one application area may be all that is needed.

Each component can have only one associated application area.

Application Area Settings and Associated Resources

Application area settings and any changes you make to these settings are automatically applied to any business component with which the application area is associated.

The resources associated with your application areas are the resources that are available to the components associated with your application area.

If resources are referenced in component steps, and you later modify the resources associated with its application area, your component may not run correctly. For example, if a component uses test objects from the **MyRepository.tsr** shared object repository, and you remove this object repository from the application area, the component will not be able to access the required test objects because the object repository is no longer included in the application area.

Tasks

How to Create and Manage Application Areas

Relevant for: GUI components only

This task includes the following steps:

- "Prerequisites " below
- "Make sure all required resources are stored in ALM" below
- "Create the application area" on the next page
- "Configure application area settings" on the next page
- "Maintain the application area" on page 2099

1. ### Prerequisites

   a. Make sure you have permissions in ALM for modifying components, adding steps, modifying steps, and deleting steps. (If you do not have all of these permissions, you can open application areas only in read-only format.) For details on setting permissions for the Business Components module, see the *HP Business Process Testing User Guide*.

   b. Evaluate the purpose of your application area and determine the resources required for the business components that will use it. For example:

      - What test objects will they need?
      - What user-defined functions can you add to ensure that all required operations are available?
      - Which keyword operations do you want to make available for each test object type?

2. ### Make sure all required resources are stored in ALM

   To create an application area, you must associate the required function library, shared object repository, and recovery scenario resources with it.

   Some of the required resources may already exist.

   If necessary, create new resources and store them in the **Test Resources** module in ALM.

   a. Verify the existence of or create required function library resources. For details, see "How to Create and Manage Function Libraries" on page 1041.

      > **Note:** BPT provides a set of predefined function libraries that you can use. Some of

> these are associated with all new application areas by default. These files are located in the **Test Resources** module of your ALM project under **Resources/BPT Resources**.

b. Verify the existence of or create required shared object repository resources. For details, see "How to Create and Manage Shared Object Repositories" on page 1291 and "How to Manage Objects in Shared Object Repositories" on page 1294.

c. Verify the existence of or create required recovery scenario resources. For details, see "How to Create and Manage Recovery Scenarios" on page 1115.

3. Create the application area

a. Connect to the ALM project in which you want to save the application area. For details, see "ALM Connection Dialog Box" on page 737.

b. Select **File > Add > New Application Area**. The New Application Area dialog box opens, enabling you to specify the location for the new application area within the ALM **Test Resources** module. (In Quality Center 10.00, application areas are always stored in the **BPT Resources\Application Areas** folder.)

   Specify a unique and descriptive name.

   After you click **Save**, the new application area is added to the open solution and then opened in the document pane. The application area displays several panes in which you specify the settings and resource files that you want business components associated with the application area to use.

> **Note:** If you do not want to add the application area to the open solution, select **File > New > Application Area**. If you do this, the open solution is closed, any open documents included in the solution are closed as well, and the new application area is opened in a new untitled solution.

4. Configure application area settings

a. In the Properties pane, provide a full description of the application area, as described in "General Properties Tab (Properties Pane for Application Areas)" on page 391.

b. Specify associations to the required UFT add-ins, as described in "General Properties Tab (Properties Pane for Application Areas)" on page 391.

   When a business process test runs, the add-ins associated with the application area for the first component in the test are loaded in UFT. Therefore, if this application area is likely to be associated with components that may be first in a business process test, ensure that you include all add-ins that any of the components in the test may need. However, to ensure expected functionality and optimum performance, do not include add-ins that are not necessary.

c. If necessary, associate function libraries or shared object repositories. Click the **<application area name>** node in the Solution Explorer and select **Associate Function Library** or **Associate Object Repository**, as needed. For more details and association methods, see "Function Libraries Pane (Application Area)" on page 2102

d. If necessary, associate recovery scenarios and define their settings as described in "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 613.

> **Note:** In general, you should create all necessary resources for your application area before you begin creating the application area itself. If you realize the need for another resource while you are in the process of creating the application area, it is possible to create new, empty resource files directly from the application area and associate them with your application area (and then create the content for those files at a later time).
>
> For details, see "Function Libraries Pane (Application Area)" on page 2102, "Object Repositories Pane (Application Area)" on page 2105, and "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 613.

e. Specify which keywords will be available for use when creating component steps, as described in "Keywords Pane (Application Area)" on page 2107.

f. Specify the Windows-based applications on which components associated with the application area can record and run, as described in "Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 597.

g. If relevant, define additional setting for your application area in the Additional Settings pane, such as add-in specific settings, log tracking, and local system monitors. For details, see "Application Area User Interface" on page 2101.

h. Save your changes 💾.

5. Maintain the application area

After you create an application area, you can use the options described below to maintain it:

- Open an existing application area for viewing or modification. For details, see "Application Area User Interface" on page 2101.

- In ALM, you can view all components that are dependent on a particular application area. To view this list, display the **Dependencies** tab for your Application Area (located in the **Test Resources** module). For details on ALM dependencies, see the *HP Application Lifecycle Management User Guide*.

- If an application area is no longer needed, you can delete it. Before you delete an application area, make sure that:
  - The application area is not associated with any component.
  - The application area is not currently open on any computer.

  You delete an application area from the Test Resources module in ALM, regardless of whether it was created in UFT or in ALM. For details, see the *HP Business Process Testing User Guide*.

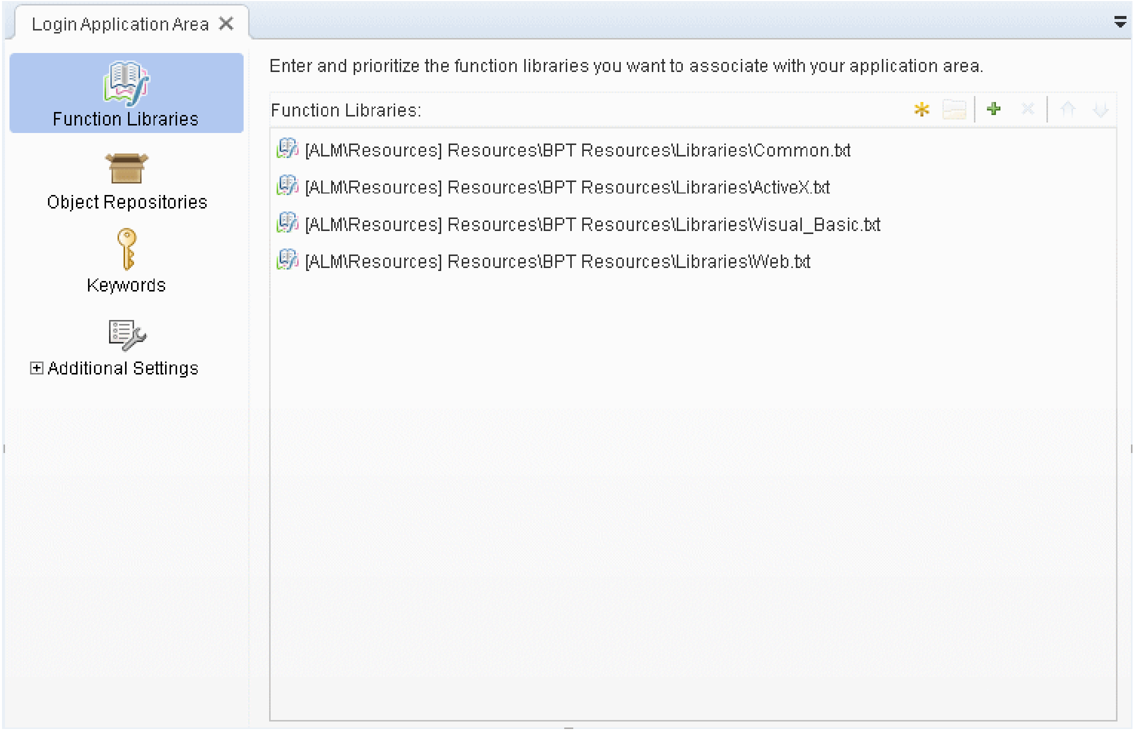
Reference

Application Area User Interface

Relevant for: GUI components only

In the application area, you define and view application area settings and associate required resource files.

Note: Some basic application area settings are defined in the "General Properties Tab (Properties Pane for Application Areas)".



Enter and prioritize the function libraries you want to associate with your application area.

[ALM\Resources] Resources\BPT Resources\Libraries\Common.txt
[ALM\Resources] Resources\BPT Resources\Libraries\ActiveX.txt
[ALM\Resources] Resources\BPT Resources\Libraries\Visual_Basic.txt
[ALM\Resources] Resources\BPT Resources\Libraries\Web.txt

To access	**In UFT:** 1. Connect to your ALM project. 2. Create a new application area or open an existing one (**File > New/Open/Add > Application Area**, or double-click the application area in the Solution Explorer). **Note:** You can also access this dialog box from ALM.
Relevant tasks	"How to Create and Manage Application Areas" on page 2097

See also	- "Application Areas - Overview" on page 2096 - "General Properties Tab (Properties Pane for Application Areas)" on page 391

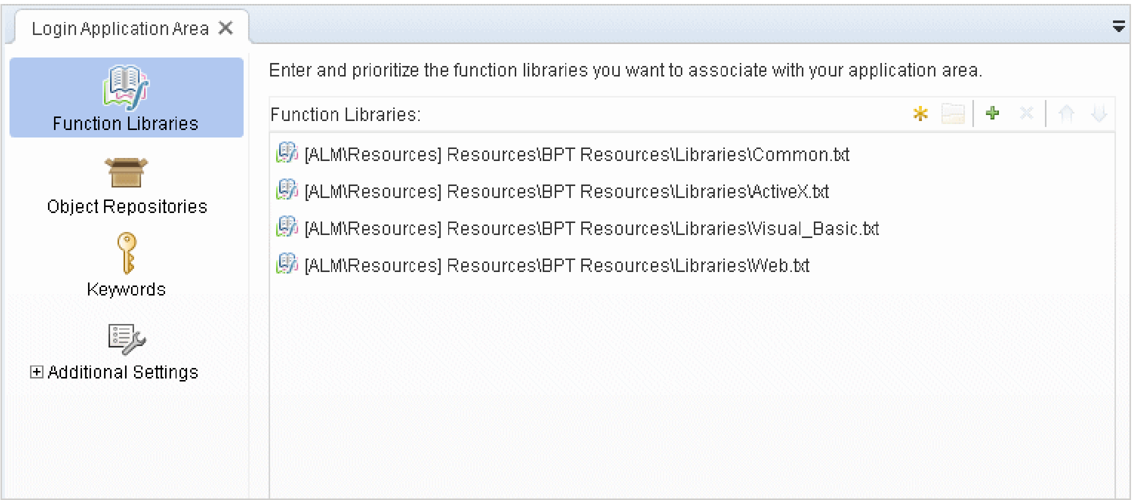
User interface elements are described below:

UI Elements	Description
	Function Libraries. Displays the Function Libraries pane, which enables you to associate function libraries with your application area and to prioritize them. It also enables you to create and modify associated function libraries. For details, see "Function Libraries Pane (Application Area)" below.
	Object Repositories. Displays the Object Repositories pane, which enables you to associate shared object repositories with your application area and to prioritize them. It also enables you to create and modify associated object repositories. For details, see "Object Repositories Pane (Application Area)" on page 2105.
	Keywords. Displays the Keywords pane, which enables you to determine which built-in and user-defined keywords (operations) are available when creating components. For details, see "Keywords Pane (Application Area)" on page 2107.
	Additional Settings. Displays the Additional Settings pane, which enables you to define settings that affect all of the components associated with this application area. You can specify the Windows-based applications with which a component can work, recovery scenarios that a component can use, and different tracking and monitoring options. For details, see "Additional Settings Pane (Application Area)" on page 2109. **Note:** The Business Components Settings dialog box is very similar to this pane, but displays most of the settings in read-only mode.

Function Libraries Pane (Application Area)

Relevant for: GUI components only

This pane enables you to associate VBScript function library files (of types .qfl, .vbs, or .txt) with your application area. You associate function libraries with your application areas to provide additional functionality in the form of user-defined keywords (functions) that can be used when creating business components. You can also create new function library files directly from this pane.



Enter and prioritize the function libraries you want to associate with your application area.

Function Libraries:

[ALM\Resources] Resources\BPT Resources\Libraries\Common.txt

[ALM\Resources] Resources\BPT Resources\Libraries\ActiveX.txt

[ALM\Resources] Resources\BPT Resources\Libraries\Visual_Basic.txt

[ALM\Resources] Resources\BPT Resources\Libraries\Web.txt

To access	In the application area sidebar, click **Function Libraries**.
Important information	• All associated function libraries must be saved in your ALM project. • To rename a function library, you can click it twice, or select it and press F2. • UFT provides you with sample function libraries containing predefined functions. By default, these files are associated with all new application areas. The default function libraries are located in the **Test Resources** module of your ALM project under **Resources/BPT Resources**. For details on creating user-defined functions in function libraries, see "User-Defined Functions and Function Libraries" on page 1029.
Relevant tasks	"How to Create and Manage Function Libraries" on page 1041
See also	"Application Area User Interface" on page 2101

User interface elements are described below:

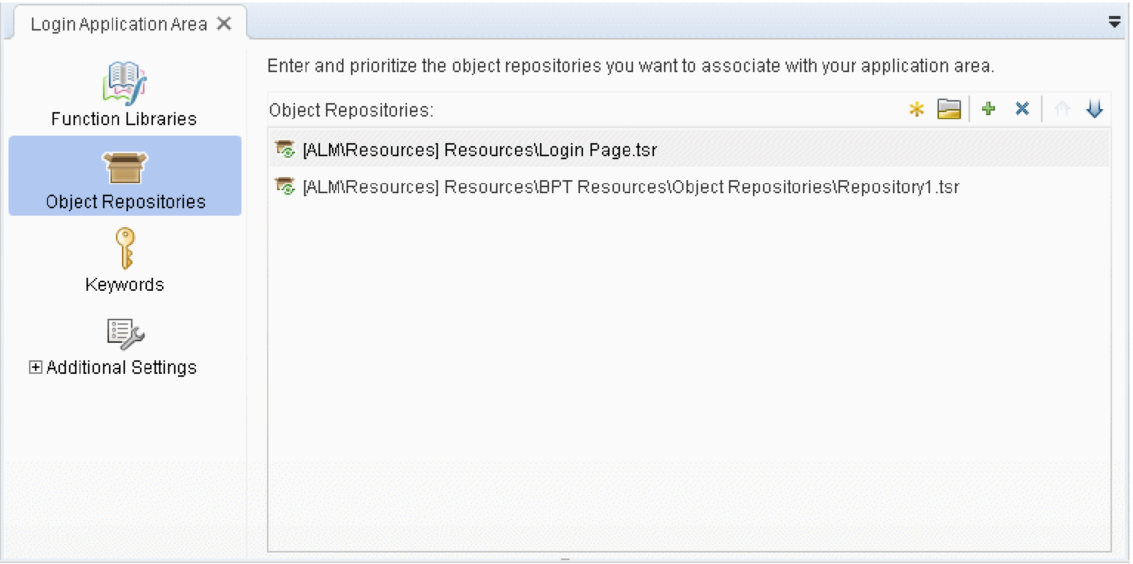
UI Element	Description
Associated function libraries	The list of function libraries currently associated with your application area. You can associate additional function libraries, and modify, delete, and prioritize these files. You can add existing function libraries or create new ones, as long as the function libraries are stored in your ALM project. **Note:** - You can double-click a function library in this list, or right-click it and select **Open** to open the function library in a separate document tab. - You can right-click a function library in this list and select **Remove** to remove its association with the application area. - If an associated function library cannot be found, for example, if it was removed from the ALM project, UFT indicates this by displaying the **Missing Function Library** 📄 icon to the left of the function library in the list. To handle the missing function library, right-click it and select **Locate** to browse to the required function library, or **Remove** to remove the association to the missing function library.
✳	Enables you to create a new, blank function library, save it to your ALM project, and add it to the list. By default, new function libraries are created as .qfl files if no suffix is specified. You can also specify a .txt or .vbs file extension.
📂	Opens the selected function library for viewing or editing in the document pane. You can also double-click a file in the list to open it. To save your changes, close the file and click **Yes** when prompted. Function libraries that are currently in use by another UFT or ALM user are locked and can be opened only in read-only mode. For details, see "User-Defined Functions and Function Libraries" on page 1029.
➕	Enables you to browse to your ALM project and select an existing function library to associate with the application area. For details, see "How to Create and Manage Application Areas" on page 2097. If the function library you select contains syntax errors, a message box informs you that components using this function library may fail because of these syntax errors.
✖	Removes the selected function library from the application area.
⬆	Moves the selected function library up in the list, giving it a higher priority during the component run session.

UI Element	Description
↓	Moves the selected function library down in the list, giving it a lower priority during the component run session.

Object Repositories Pane (Application Area)

Relevant for: GUI components only

This pane displays the list of shared object repositories currently associated with your application area and enables you to associate additional object repositories, and to modify, delete, and prioritize these files. You can also create new shared object repositories directly from this pane.



Enter and prioritize the object repositories you want to associate with your application area.

[ALM\Resources] Resources\Login Page.tsr

[ALM\Resources] Resources\BPT Resources\Object Repositories\Repository1.tsr

To access	In the application area sidebar, click **Object Repositories**.
Relevant tasks	• "How to Create and Manage Shared Object Repositories" on page 1291 • "How to Manage Objects in Shared Object Repositories" on page 1294
See also	• "Application Area User Interface" on page 2101 • "Shared Object Repositories" on page 1285 • "Renaming Test Objects " on page 1206

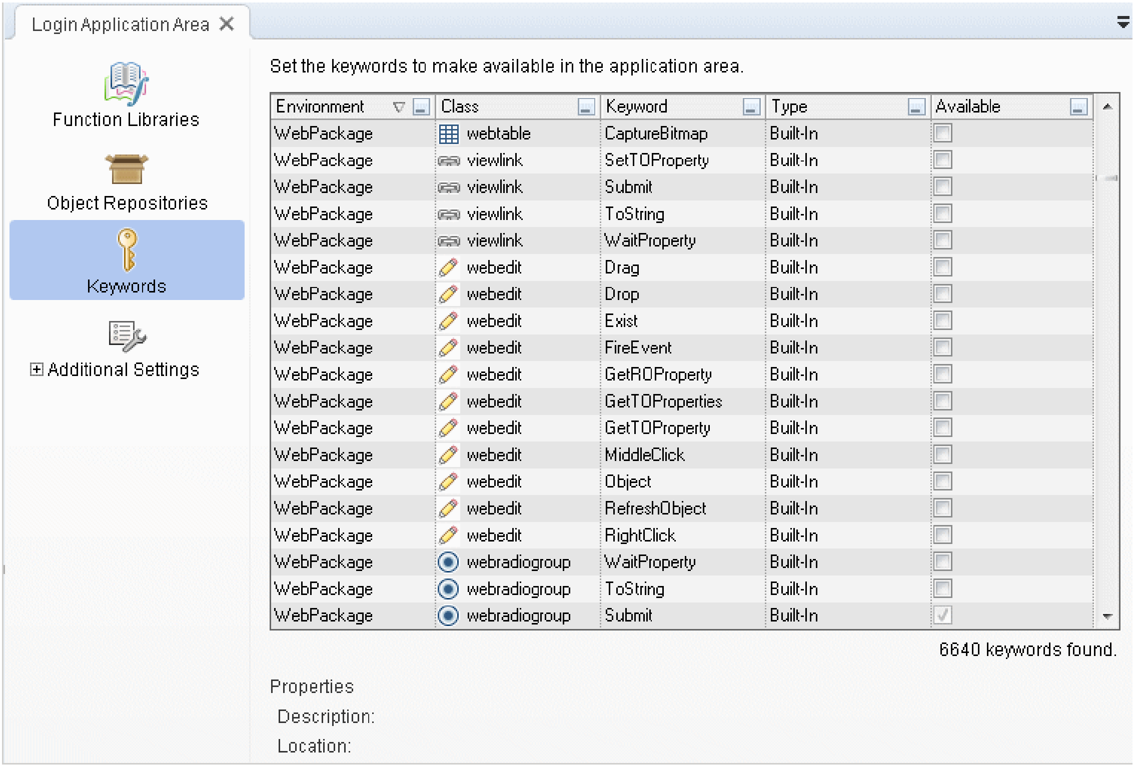
User interface elements are described below:

UI Element	Description
Associate object repositories	The list of object repositories that are currently associated with your application area. You can add existing object repositories or create new ones, as long as the object repositories are stored in your ALM project. All business components associated with an application area that refers to these shared object repositories will then access these shared object repository files. **Note:** • You can double-click a shared object repository in this list, or right-click it and select **Open** to open it in the Object Repository Manager. • You can right-click a shared object repository and select **Remove** to remove its association with the application area. • If a shared object repository cannot be found, its name and path are displayed in the Errors pane when you open the application area. To handle the missing shared object repository, right-click it in the list of associated object repositories and select **Locate** to browse to the required shared object repository, or **Remove** to remove the association to the shared object repository.
✱	Enables you to create a new object repository, save it to ALM, and then add it to the list. For details, see "How to Create and Manage Application Areas" on page 2097.
📂	Opens the selected object repository for viewing or editing in the Object Repository Manager. You can also double-click an object repository in the list to open it. Object repositories that are currently locked are opened in read-only format. For more information on the Object Repository Manager, see "Shared Object Repositories" on page 1285.
⊞	Enables you to browse to the Test Plan tree or Resources module of your ALM project and select an existing object repository to associate with the application area. For details, see "How to Create and Manage Application Areas" on page 2097.
☒	Removes the selected object repository from the application area.
↑	Moves the selected object repository up in the list, giving it a higher priority during the component run session.
↓	Moves the selected object repository down in the list, giving it a lower priority during the component run session.

Keywords Pane (Application Area)

Relevant for: Keyword GUI components only

This pane enables you to manage keywords and select which of them should be available to as test object operations when creating keyword components. Only **selected** built-in keywords are available by default. However, all user-defined keywords (functions) are available.



Set the keywords to make available in the application area.

Environment	Class	Keyword	Type	Available
WebPackage	webtable	CaptureBitmap	Built-In	□
WebPackage	viewlink	SetTOProperty	Built-In	□
WebPackage	viewlink	Submit	Built-In	□
WebPackage	viewlink	ToString	Built-In	□
WebPackage	viewlink	WaitProperty	Built-In	□
WebPackage	webedit	Drag	Built-In	□
WebPackage	webedit	Drop	Built-In	□
WebPackage	webedit	Exist	Built-In	□
WebPackage	webedit	FireEvent	Built-In	□
WebPackage	webedit	GetROProperty	Built-In	□
WebPackage	webedit	GetTOProperties	Built-In	□
WebPackage	webedit	GetTOProperty	Built-In	□
WebPackage	webedit	MiddleClick	Built-In	□
WebPackage	webedit	Object	Built-In	□
WebPackage	webedit	RefreshObject	Built-In	□
WebPackage	webedit	RightClick	Built-In	□
WebPackage	webradiogroup	WaitProperty	Built-In	□
WebPackage	webradiogroup	ToString	Built-In	□
WebPackage	webradiogroup	Submit	Built-In	☑

6640 keywords found.

Properties

Description:

Location:

To access	In the application area sidebar, click **Keywords**.
Important information	• All of the built-in methods and properties, plus all of the functions in user-defined function libraries, are displayed as keywords in the Keywords pane. In addition, the Keywords pane displays methods and properties of any test object classes that you or a third party developed using UFT Add-in Extensibility. • The Keywords pane is not relevant for scripted components. • You can also filter and sort columns. For details, see "Filtering and Sorting Options" on the next page.
See also	"Application Area User Interface" on page 2101

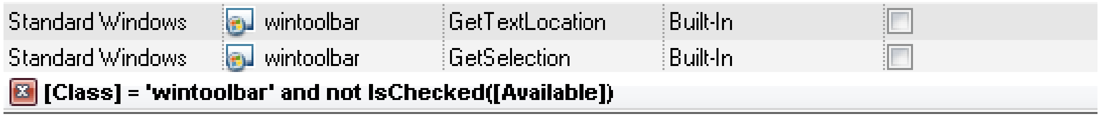
User interface elements are described below:

Column	Description
Environment	The name of the add-in for which the keyword is provided, for example, Web or Visual Basic. The keywords available for all currently loaded add-ins are displayed in the pane. (Including keywords and add-ins that you or a third party developed using UFT Add-in Extensibility.) • Keywords in user-defined functions that are registered to a test object are displayed under the environment and object class to which they are registered. • Keywords in user-defined functions that are not registered to a test object, plus built-in VBScript functions, are all displayed under the **Global** environment.
Class	The object class, for example, Image or Winbutton.
Keyword	The displayed operation name, for example, Click or VerifyProperty.
Type	Indicates whether the operation is Built-in (provided by UFT) or User-Defined (contained in a function library).
Available	Indicates whether the keyword is available for use in business component steps. You can select or clear each check box as required.
Properties	Textual description of what the keyword does, and the name and path of its function library (for user-defined keywords). The location of a built-in keyword is defined as Internal.
⊟	**Filter.** Displays a list of the unique items contained in the selected column. You can filter the data in the column to display only those keywords with which you want to work. You can filter the data in a single column only, or filter additional columns to further reduce the number of displayed items. For example, you may want to view only Web Add-in keywords that are currently not available. You would filter the **Environment** column to display only keywords from the Web Add-in, and then filter the **Available** column to display only keywords whose check box is cleared (select **Unchecked** from the **Available** column filter list). For details, see "Filtering and Sorting Options" below.

Filtering and Sorting Options

- Click an item in the list. You can use the **CTRL** key to select multiple items from a filter list. The Keywords pane refreshes to show the data for keywords with that item name only.

  You can then click the arrow in another column header and select an item in that list. The filtered data is filtered again to show only the keywords that match all selected filter criteria.



Standard Windows	wintoolbar	GetTextLocation	Built-In	□
Standard Windows	wintoolbar	GetSelection	Built-In	□

☒ **[Class] = 'wintoolbar' and not IsChecked([Available])**

- In the **Filter For** box at the bottom of the filter list, you can enter a filter pattern that includes wildcards such as **?**, *****, and **#**. Press **ENTER** to filter the data according to the pattern. You can use **?** to represent any single character, ***** to represent zero or more occurrences of any character, and **#** to represent any digit. You can also use **|** to specify items that match only one of the options in the pattern. For example, Verify*|Check* shows all keywords that start with **Verify** or **Check**.

- You can apply multiple filters simultaneously. For example, if you want to view keywords for only the Standard Windows and ActiveX environments, and you want to display only built-in keywords (as opposed to user-defined keywords), you can apply three filters: one filter for StandardWindows; another filter for ActiveX; and a third filter for the type, Built-in.

- Click the ☒ to the left of the filter criteria to clear the filter and show all keywords.

- You can rearrange the order that columns are displayed in the Keywords pane by dragging a column header to a new location. Red arrows are displayed when the column is dragged to an available location.

- If the data in a column is partially hidden because the column is too narrow, you can resize the column using the mouse. Drag a column header divider to adjust the width.

- You can arrange the data in a column into ascending or descending alphabetical order by clicking the column header. The **Available** column is sorted according to selected and cleared check boxes.

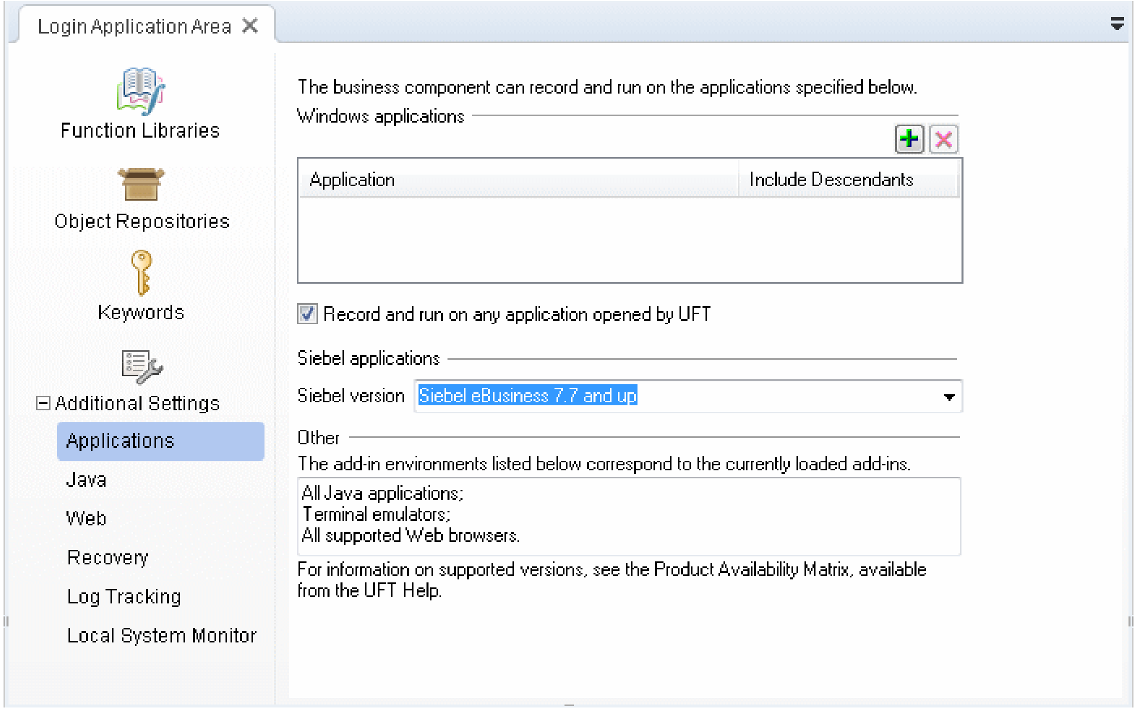
- The sort direction is indicated by an arrow in the column header. Click the column header again to sort the data in the other direction.

Additional Settings Pane (Application Area)

Relevant for: GUI components only

This pane enables you to define settings that affect all of the components associated with this application area.

> **Note:** The Business Components Settings dialog box is very similar to this pane, but displays most of the settings in read-only mode.



The business component can record and run on the applications specified below.

Windows applications

Application	Include Descendants

☑ Record and run on any application opened by UFT

Siebel applications

Siebel version: Siebel eBusiness 7.7 and up

Other

The add-in environments listed below correspond to the currently loaded add-ins.

All Java applications;
Terminal emulators;
All supported Web browsers.

For information on supported versions, see the Product Availability Matrix, available from the UFT Help.

To access	In the application area sidebar, click **Additional Settings**.
Important information	This pane may also contain additional panes corresponding to any UFT add-ins that are loaded, for example, Web, or Java. For details on nodes related to add-ins, see the *HP Unified Functional Testing Add-ins Guide*.
See also	• "Application Area User Interface" on page 2101 • "Applications Pane (Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 597 • "Recovery Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 613 • "Log Tracking" on page 586 • "Local System Monitor Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 623

The Additional Settings pane contains a number of sub-panes that enable you to define the following settings:

Pane	Enables you to define:
Applications	The Windows-based applications with which a component associated with your application area can work.

Pane	Enables you to define:
Recovery	Recovery scenarios that specify how a component associated with the application area recovers from unexpected events and errors during a run session.
Log Tracking	Run-time preferences for tracking log messages generated by the log framework that monitors events occurring in your application.
Local System Monitors	Preferences for tracking system counters during a run session.

Troubleshooting and Limitations - Application Areas

Relevant for: GUI components only

This section describes troubleshooting and limitations for application areas.

- Renaming a UFT test or component from ALM may cause the test or component to behave unexpectedly.

  Workaround: To rename a test or component, open it in UFT and rename it using the **Save As** option. If the test or component has already been renamed from ALM, use the rename option again to restore the old name, and then use the **Save As** option in UFT.

- Renaming a UFT test parameter from UFT will cause any run-time parameter values already set for this parameter in ALM to be lost. For a list of naming conventions, see "Troubleshooting - Naming Conventions" on page 2269.

Chapter 67: Using Data in Business Process Testing

Relevant for: business process testing

This chapter includes:

Concepts

Using Data in Business Process Tests - Overview

Relevant for: business process testing

You can affect the behavior and results of a business process test by using parameters to define the values that components and flows receive and return. This process is known as **parameterization**.

> **Note:** You can also use parameters for individual steps inside a component. For details, see "Parameterizing Object Values" on page 1526.

Parameterization enables you to perform operations on the application you are testing with multiple sets of data. Each time you run a business process test, you can supply different values for the parameters in the test (or its components and flows).

For task details, see "How to Use Data in a Business Process Test" on page 2124.

This section includes:

- "Parameter Types" below
- "Parameter Linkage" on the next page
- "Parameter Promotion" on page 2116
- "Parameters and Iterations" on page 2116

Parameter Types

Business Process Testing provides several types of parameters, such as

- **Component/Test/Flow parameters.** Component parameters provide data for components. Similarly, flow parameters provide data at the flow level and test parameters provide data at the test level.
- Input/Output parameters. Input parameters are data that are used by an entity (component, flow, or test) in order for the entity to perform its function. For example, in order for a component to simulate a login operation, it must receive the login name and password as input parameters. Output parameters are data that are outputted or generated by an entity (component or flow), such as an invoice number.

> **Example**
>
> To test the business process of a banker logging into an online banking application, you might structure a business process test from components that:

- Log into the application (Login)
- Select a customer loan (SelectLoan)
- View transactions for the loan (ViewLoan)
- Log out (Logout)

You can set up the steps in each of these business components to receive data from the business process test that runs the components (for example, the loans that a customer has). You can also parameterize any element of data, which may have different values each time the business component is run. For example, the banker may choose a different customer and customer loan to view each time he or she logs in.

Here are parameters that you might create for this scenario, listed by category:

Category	Parameters
Input parameters for a component	• LoginName, entered as input by the banker when logging in • AccountNo, entered by the banker, perhaps from a written inquiry. These parameters are created as input parameters for an individual component, and then you can use them for any step inside the component.
Output parameters for a component	• SessionNo, a number for the login session, outputted by the business component when the banker successfully logs in • SelectedAccountNo, outputted by the business component after the banker selects a loan from a list
Test Parameters	CustomerLoans, a comma-delimited list of all loans for a particular customer, accessed from the test level.

Parameter Linkage

For component parameters within a business process test or flow to be accessible to other components, the parameters must be linked. You link output parameters from one component to input parameters in another component.

Tip: You can also link flow parameters.

For task details, see "Link parameters" on page 2125.

Example

A banking application contains business processes for:

- Selecting a customer loan (SelectLoan)
- Viewing transactions for the loan (ViewLoan)

You can structure your business process test so that it contains a component for selecting the loan (called SelectLoan), and component for viewing the loan's transactions (called ViewLoan). For the ViewLoan component to know which loan to view, it receives an input parameter (called ViewLoanID) from the output parameter (called SelectedLoanID of the SelectLoan component.

Parameter Promotion

Promoting parameters enables in the components in other flows or components and flows in other tests to access a parameter value. Promoting a parameter from the component level wides the scope of a parameter so that more entities can use its value when a business process test runs. By promoting a parameter from the component or flow level, you make it into a flow or test parameter and then any component included in the flow or test can access the promoted parameter.

You can promote component parameters to the flow or test level at the same time as you add a component to a flow or test. Similarly, you can promote flow parameters to the test level at the same time that you add a flow to a test.

For task details, see "Promote parameters" on page 2125.

Example

Using the same banking application, you decide that once a loan ID is selected by a banker using the application, all components testing the application included in a specific flow or test should have access to the loan ID parameter (called SelectedLoanID).

To make the SelectedLoanID output parameter available to all components in a flow, you promote it from the component or flow to the flow or test level. Once promoted, any business components in the same business process test can use the SelectedLoanID parameter as an input parameter. These components would then have access to the SelectedLoanID parameter without needing to directly link input and output parameters

Parameters and Iterations

Setting parameter values per test iteration enables you to iterate specific components in a business process test, specific components in a flow, or entire business process tests--thereby creating data-driven tests.

- Defining iterations enables you to automatically run business components, flows, and tests multiple times, each time using different parameter values. For details, see "Running Iterations in a Business Process Test" on page 2132.
- You can run business process tests for different use-cases of the application you are testing. Examples of use-cases include running the test on different operating systems, running the same test for different browser versions, or running the same test with different languages or character sets.

> **Note:** In ALM, you can associate each of these use-cases with a test configuration. For details, see the *HP Application Lifecycle Management User Guide*.

Example

You must test that the banking application business processes for approving loans works as expected for different scenarios:

- Loans pre-approved during the last marketing campaign are automatically approved
- Loans under a certain amount follow the standard business process for approving loans
- Loans over a certain amount must be flagged for additional approvals

To ensure these business processes work correctly, you can iterate a loan approval component contained in your test (named ApproveLoan). In each of the iterations of this component, you provide different values for the component parameters set up for the loan amount (called LoanAmount) and for the date of the marketing campaign (called PromotionCode).

Business Process Testing Parameter Categories

Relevant for: business process testing

Within a BPT test, you can have different types of parameters:

Parameter Categories	Parameter and Description

Input / Output	**Input parameters** enable you to define data used by a component or flow that is provided from an external source. When creating components, tests, and flows, you define how the values are supplied for input parameters. An input parameter can receive: • A predefined default value, if no other value is supplied by the test or flow. • An output parameter value returned by a component or flow earlier in the flow or test. • A parameter value that is supplied at the test or flow level, when the test or flow is run. **Output parameters** allow data values retrieved from a component step or flow (the source) to be passed as input parameters to a subsequent component or flow (the target) in the test run. **Note:** You cannot set a default value for an output parameter. You can link Input and output parameters to make data available between components or flows within the same business process test.
Component, Flow, and Test	**Component parameters** are parameters defined with a component, and available for use for all steps included in the component. You can have both component input parameters and component output parameters. These parameters are available to: • All subsequent steps in the same component • Subsequent components in a flow or test, provided that: ▪ The component parameter is defined as an output parameter in the current test or flow, and as an input parameter in the subsequent component in the current test or flow. ▪ The output parameter in the current test or flow is linked to the input parameter of the subsequent component in the current test or flow. **Flow parameters** are parameters defined within a flow. These parameters are available to all components included in a flow. Like component parameters, flow parameters can be input or output parameters. **Test parameters** are parameters identified within a business process test. These parameters are available to all components and flows in the test. Test parameters can only be input parameters.

Local parameters	**Local parameter** values are defined within an individual business component and you can access them only in the component in which they were created. Local parameters are intended for use in a single step or between component steps, for example, as an output value for one step and input parameter for a later step. This type of parameter is used when working with keyword GUI components.

Linking Parameters in Business Process Tests

Relevant for: business process testing

Parameter linkage enables you to pass data between business components and flows. Thus, if you have a value that you need for another component in one business component, you can pass the value between the components by linking their parameters. This tests the ability of your application to pass a value from one API to another in the course of the application's work.

You can also pass a test or flow parameter to an individual component parameter.

Once a parameter is linked, it is available to any step contained in the component with the parameter.

Example

A business component,(named CreateLoan) has an output parameter that contains a generated loan ID. A subsequent business component, (named SearchLoan), can verify the loan if it has access to the CreateLoan loan ID value. This access is provided by linking the CreateLoan component output parameter to SearchLoan component input parameter.

The component or flow in which the output parameter is defined is the source. The component or flow that links to that output parameter is the target. In the example above, CreateLoan is the source component and SearchLoan is the target component.

For task details, see "Link parameters" on page 2125.

This section also includes:

Linking Parameters When Running Iterations in a Business Process Test

Relevant for: business process testing

As part of data driving a business process test, you can set components (or groups of components) to run multiple times in different iterations. For details on iterations, see "Running Iterations in a Business Process Test" on page 2132.

Iterations of a source component can result in multiple parameter output values. In these cases, the value provided by each iteration is passed as input to the corresponding iteration of the target.

Linking can occur successfully only when UFT can determine the target iteration for each source iteration. One of the following conditions must exist:

- **Condition 1.** The source has one iteration and the target has one or more iterations (a “1–to–n”relationship). For an example, see "Linking Parameters when using Iterations ("1-to-n" Relationship)" on page 2122.

- **Condition 2.** The source and the target have the same number of iterations (an “n–to–n" relationship). For an example, see "Linking Parameters when using Iterations ("n-to-n" Relationship)" on page 2122.

> **Note:** When a source or target is a member of a group, the number of iterations is that of the group.

If the component iterations are not represented by a “1–to–n” or “n–to–n” relationship, a warning message is displayed.

Considerations

- When you use the output parameter of a previous component as the value for an input parameter of a subsequent component, the link between the output and input parameter applies to all component iterations of the input parameter.

- When iterations of a source component in a business process test result in multiple output parameter values, the value that is provided by a given iteration run is passed as input to the corresponding iteration of the target component.

- Moving a business component, group, or flow can cause a parameter reference conflict, such as when a target component is moved to a position preceding the source component. If the resulting warning message is ignored, the conflicting link to the source parameter is deleted. This causes the iteration to fail, and you must redo the link between the parameters.

- Iteration errors cause a business process test or flow that contains the relevant components to fail. These errors are reported as part of the test results.

Linking Parameters in a Business Process Test - Examples

Relevant for: business process testing

When linking component parameters as part of a business process test, there are a number of possible scenarios:

This topic includes the following examples:

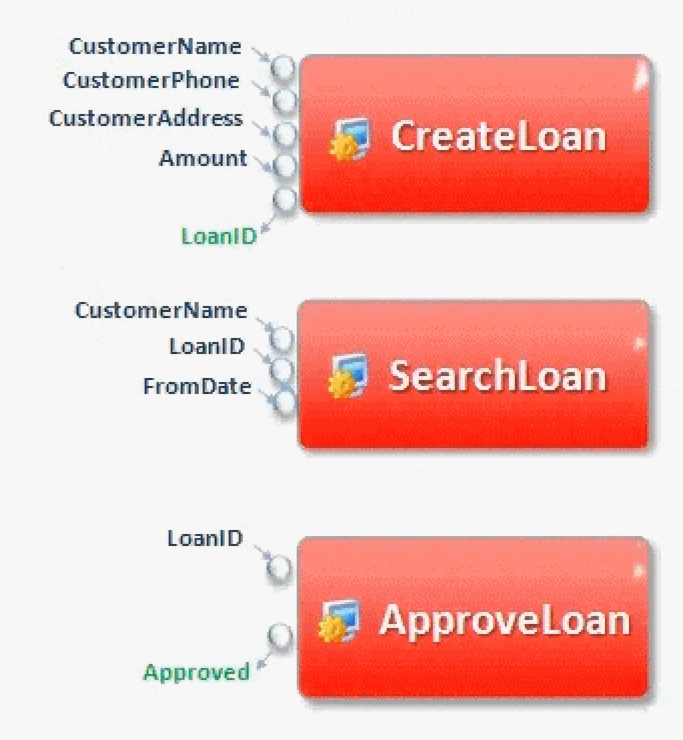
- "Linking Input and Output Parameters" on the next page

- "Linking Parameters when using Iterations ("1-to-n" Relationship)" on page 2122

- "Linking Parameters when using Iterations ("n-to-n" Relationship)" on the next page

For these examples, you create components corresponding to the different processes of your application for processing a customer loan request:

- A component that tests the application's ability to receive the request, and generate a unique loan ID for the request (called CreateLoan). This
- A component that searches the existing loans to verify if the loan already exists (called SearchLoan).
- A component that tests the loan request approval process (called ApproveLoan).



Each of the components has a number of input parameters (in black), and output parameters (in green).

Linking Input and Output Parameters

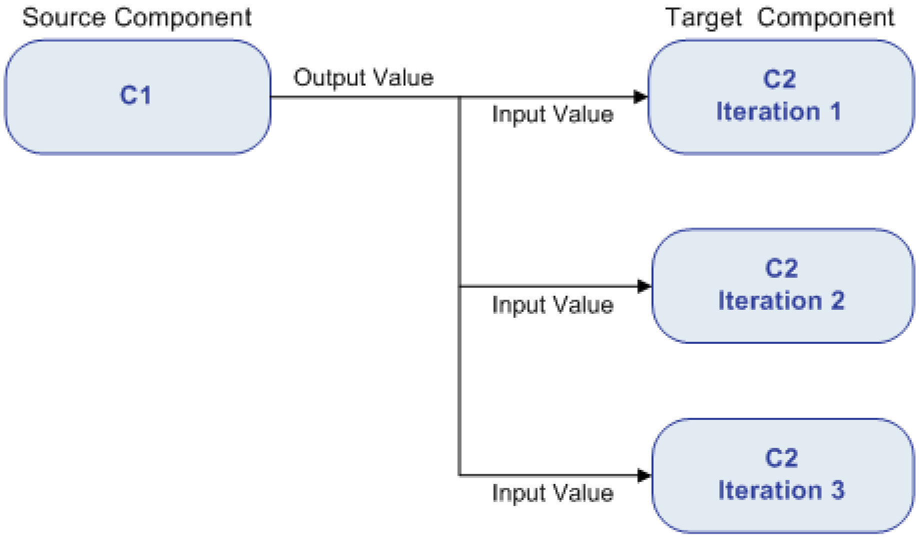
When you create a business process test, you arrange the components in order to test the entire loan approval process workflow from receiving the request through approving the request.

In this example, the value of the LoanID output parameter is passed from the CreateLoan component to the SearchLoan component as the value of the LoanID input parameter for the component. The value is also passed to the ApproveLoan component as the value for the LoanID input parameter.

Linking Parameters when using Iterations ("1-to-n" Relationship)

In this instance (using the example above), the CreateLoan component (containing the LoanID output parameter) has one iteration, and the SearchLoan and ApproveLoan components have one or more iterations. This is called a **"1-to-n"** relationship.

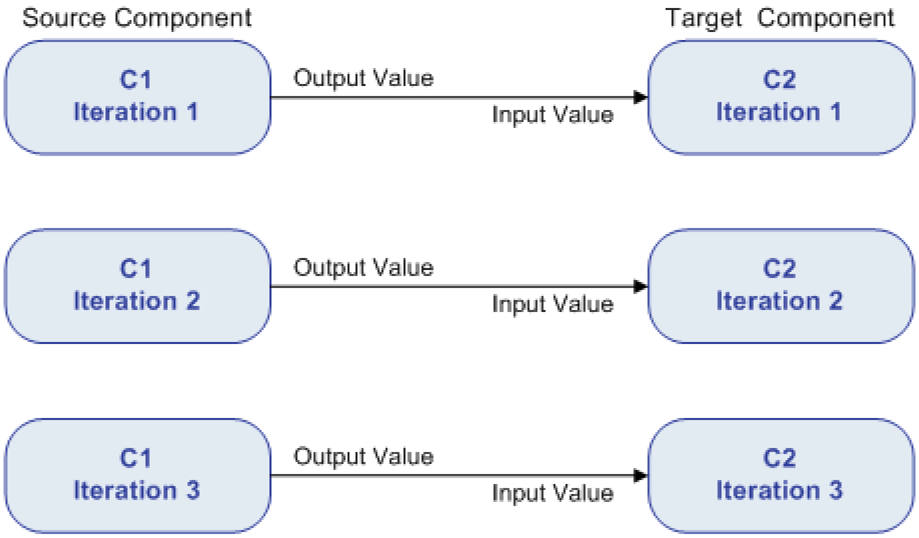
In this scenario, the value of the LoanID output parameter is used for each iteration of the SearchLoan or ApproveLoan component:



Linking Parameters when using Iterations ("n-to-n" Relationship)

In this scenario, the CreateLoan component (containing the LoanID output parameter) has the same number of iterations as the SearchLoan and ApproveLoan components. This is called an **"n-to-n"** relationship.

For this scenario, the different values for the LoanID output parameter in each of the iterations are used in the respective iterations of the SearchLoan or ApproveLoan components:



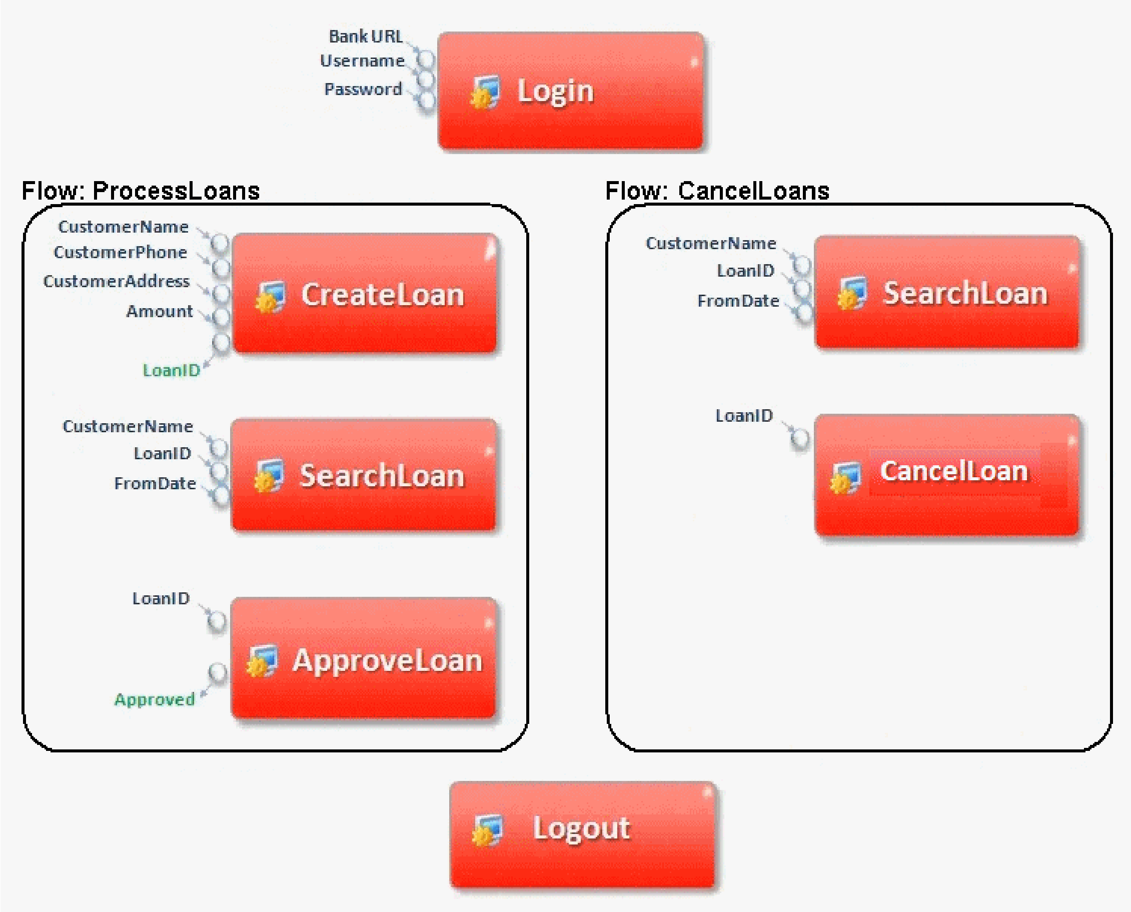
Promoting Parameters in a Business Process Test

Relevant for: business process testing

When editing a business process test, you can promote component parameters to the flow or test level at the same time as you add a component to a flow or test. Similarly, you can promote flow parameters to the test level at the same time as you add a flow to a test. This enables you to use a parameter for all components or flows included in a given flow or test.

Example

You create a business process test with components named CreateLoan, VerifyLoan, and ApproveLoan. Each of these three components contains a parameter called LoanID:



Using parameter promotion, you can promote the LoanID parameter to the ProcessLoans flow level, thereby making it available as a parameter for the CreateLoan, SearchLoan, and ApproveLoan components. Likewise, if you need to make the LoanID parameter available to the CancelLoans flow, you can promote it to the test level.

For task information, see "Promote parameters" on page 2125.

Tasks

How to Use Data in a Business Process Test

Relevant for: business process testing

This task provides general information for how to work with parameters, and iterations in business process tests.

This task is part of a higher-level task. For details, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

This task includes the following steps:

- "Design data" below
- "Create parameters and set default values" below
- "Link parameters" on the next page
- "Promote parameters" on the next page
- "Add iterations for a component or flow" on page 2126
- "Set data values for each iteration" on page 2126

Design data

Consider the following before using parameters:

- Determine which parameters are dependent on other parameters so that you can link them. For details, see "Link parameters" on the next page.
- Determine which parameters should be available at the component, flow, and test levels. For details, see "Promoting Parameters in a Business Process Test" on the previous page.
- You can also use different use-case scenarios by creating iterations for a given component, flow, or test and providing values for the parameter in each iteration. Design how may times each component, flow, and business process test should run, and with what values. For conceptual details, see "Running Iterations in a Business Process Test" on page 2132.

Create parameters and set default values

1. Prerequisite: Ensure that you add the component or flow to your open solution and check it out of the version control database (if your components/flows are saved in a version-controlled ALM project.)
2. In the Solution Explorer, select the test, flow, or component to which you want to add a parameter.

3. In the Properties pane, open the **Test Parameters** or **Parameters** tab.

4. In the Test Parameters/Parameters, tab, click the **Add** button and specify the type of parameter to add (either input or output).

5. In the Add Input Parameter/Add Output Parameter dialog box, enter the parameter details. You can enter a default value or leave the field blank.

   If you enter a default value, you can use the default value in the event a value is not supplied for the test run, or you can use the default value as an example for the type of value that can be provided (for example, a phone number example could be ###-##-####).

Note: After you add a component parameter, this parameter is available to use in any step contained in a component. For details on parameterizing component steps, see "Parameterizing Object Values" on page 1526.

Link parameters

1. Prerequisite: Ensure that you add the component or flow to your open solution and check it out of the version control database (if your components/flows are saved in a version-controlled ALM project.)

2. In the document pane, select a business process test or flow.

3. In the business process test or flow tab, select a flow or component.

4. In the Properties pane, open the **Test Parameters** or **Parameters** tab.

5. Select the parameter you want to link.

6. In the Value cell of the parameter, click the **Link** button. The Select Link Source dialog box opens.

7. In the left pane of the Select Link Source dialog box, select the test or flow from which you want to use a parameter.

8. In the right pane of the Select link Source dialog box, select the parameter to which to link.

   The parameter name is displayed with a link icon in the value column indicating the link.

Tip: You can also automatically link component parameters if they share the same name as a test or flow parameter. In the BPT Testing tab of the Options dialog box (**Tools > Options > BPT Testing** tab), select the **Always link to existing test parameters** option.

Promote parameters

1. Open the BPT Testing tab (**Tools > Options > BPT Testing** tab).

2. In the **BPT Testing** tab, select the **Promote component parameters** option. Any parameters in components or flows added after this option is selected are promoted to test or flow parameters.

   > **Tip:** If you do not want to automatically promote parameters for a selected component or flow, clear this option before adding the flow or component.

3. If you do not automatically promote parameters using the options in the BPT Testing pane of the Options dialog box, you can click the **Promote Parameters** button in the Parameters tab of the Properties pane.

   The selected parameter is promoted to the next level and the parameter value is linked to the value of the newly created parameter.

Add iterations for a component or flow

Before you can set parameter values for each iteration of a component or flow, you must add iterations to the test for each component you want to run in iterations:

1. In the document pane, select the component or component group for which to add iterations.

2. If necessary, open the Data pane.

3. In the Data pane, click the **Add** button and select one of the following to add an iteration:

   - **Add New Iteration:** Adds a new iteration without any values for the component/group parameters.

   - **Copy Iteration.** Copies the values for the component/group parameters from the previously entered iteration.

   - **Create Iteration with Default Values**. Adds a new iteration with the default values entered for the parameter in the Test Parameters/Parameters tab in the Properties pane.

   The new iteration and parameter values (if selected) are added as an additional row in the Data pane.

Set data values for each iteration

1. In the document pane, select a flow, component, or group.

2. If necessary, open the Data pane.

3. In the Data pane, click in the **Value** cell for the parameter name. All parameters for the selected flow, component, or group are displayed in separate columns with the parameter name as column headers.

> **Note:** Make sure that you have also selected the correct row for the parameter - each separate row is its own iteration.

4. Enter the parameter value for the selected parameter.

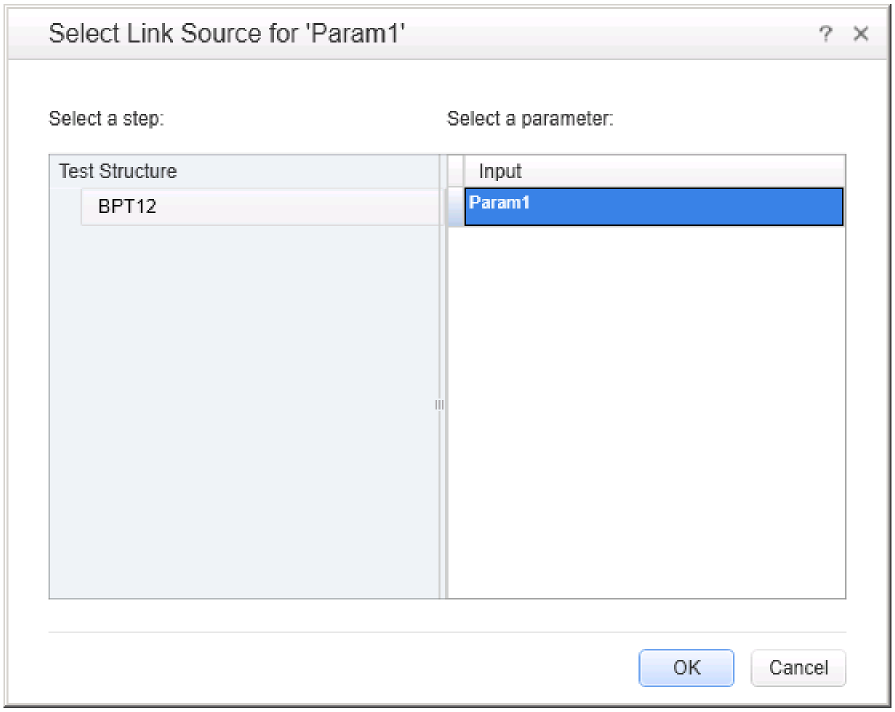
   The entered parameter value is used for each iteration.

Reference

Select Link Source Dialog Box (BPT)

Relevant for: business process tests and flows

This dialog box enables you to link parameter values to other parameters in your business process test or flow, and is accessible from the BPT Properties pane Parameters tab or the BPT Data pane.



To access	1. Make sure that you have a business process test or flow open in the document pane, with a specific component or flow selected. 2. Do one of the following: - In the Properties pane Parameters tab, hover over the **Value** column for the parameter you want to link, and then click the **Link** button. - In the Data pane, click in the cell for the parameter value you want to link, and click the **Link** button.

Important information	In the **Select a step:** column, select the test, flow, or component that contains the source parameter, and then in the **Select a parameter:** column, select the parameter you want to link to. • To link to test parameters, select the test node in the **Select a step:** column. • To link to component or flow parameters, expand the tree in the **Select a step:** column and select the component or flow node that contains your parameter. This is only available when linking from the Parameters tab of the Properties pane.
Relevant tasks	"How to Use Data in a Business Process Test" on page 2124
See Also	"Using Data in Business Process Tests - Overview" on page 2114 "Business Process Testing Parameter Categories" on page 2117 "Linking Parameters in Business Process Tests" on page 2119

User interface elements are described below:

UI Element	Description
Select a step:	Displays the current test structure available. When linking from the Data pane, this list will include tests only, as you can only link to test parameters from the Data pane. When linking from the Parameters tab in the Properties pane, this column also lists any flows or components under the relevant test, and you can link to parameters stored in those components or flows as well.
Select a parameter:	Lists the available parameters in the selected component or flow.

Troubleshooting and Limitations - Using Data with BPT in UFT

Relevant for: business process testing

- When referencing data table sheets from a business process test, you should use the data table sheet names, not the SheetIDs.

- To import a data sheet to a business component, you should use the following syntax: datatable.ImportSheet "<path to data sheet>",<row index>,Environment("ActionName")

Chapter 68: Running Business Process Tests

Relevant for: business process testing

This chapter includes:

Concepts

Running Iterations in a Business Process Test

Relevant for: business process testing

When deciding how to run your business process test, you can vary the number of times a selected flow, component group, or individual component runs in the course of the test run. This enables you to run flows or components with different sets of data.

You can configure how many times, and with which data:

- A business component runs in a test
- A flow runs in a test
- A component group runs in a test

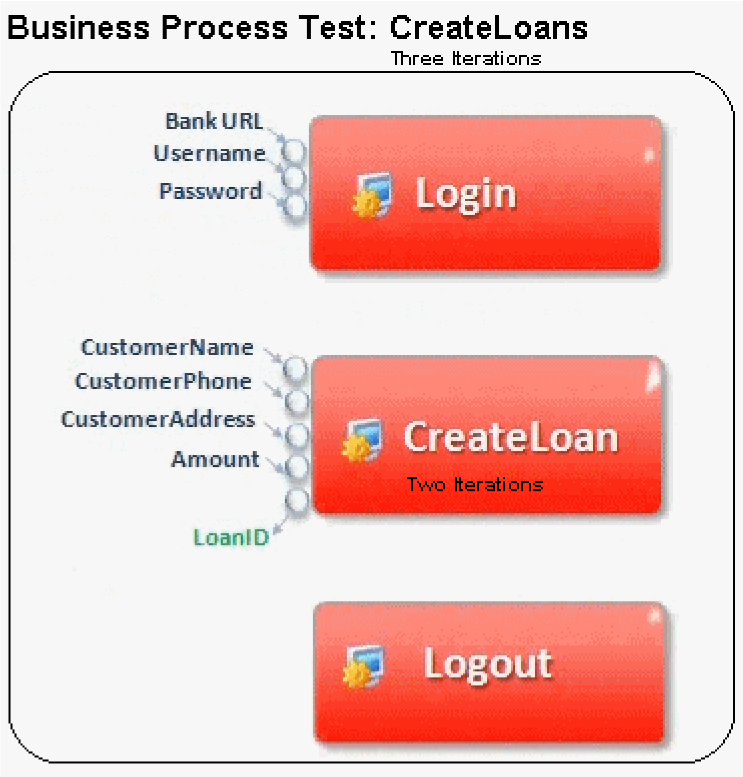
Examples

- You can create iterations for a flow that searches for different loans in a test for a banking application by supplying different loan IDs for a loan ID parameter.
- You can create iterations for a test that prepares loans with different interest rates to see which rate is most affordable for the customer.

You define the number of iterations to add in the Data pane. Then, for each iteration, you supply data for the parameter values. You can keep the default value, use the same value as a previous iteration, or vary the values between iterations.

Example

You create a business process test with components named Login, CreateLoan, and Logout.



In this scenario:

- The entire business process test is iterated three times.
- You can use different values for the parameters BankURL, Username, and Password in each test iteration.
- Within each of the three test iterations, the CreateLoan component is iterated twice. This means that the CreateLoan component iterates a total of six times.
- Different values for the CustomerName, CustomerPhone, CustomerAddress, and the Amount input parameters are used for each iteration of the CreateLoan component. Six different input parameters can be supplied in total.
- The CreateLoan component provides an output value for the LoanID parameter for each iteration (six output values provided in total).

For task details, see "Set data values for each iteration" on page 2126.

Running Iterations of Component Groups in a Business Process Test

Relevant for: business process testing

Sometimes, it may be helpful to run a group of components together for multiple iterations.

Component groups contained in your test flow are identified by a group node listed above its member components. This group node contains the group icon ▦ and displays the number of iterations for the components included in the group.

You cannot run any of the components included in the group for a different number of iterations than the number of iterations defined for the group.

Example

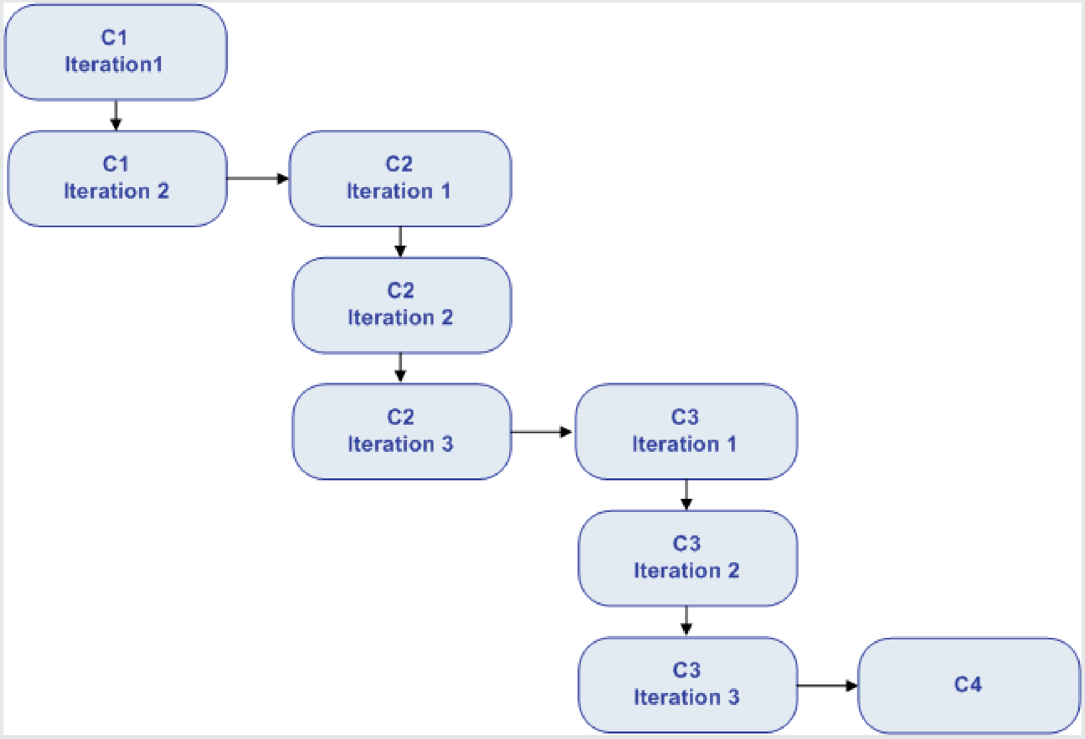
You create a business process test with the following business components: C1, C2, C3, and C4. You set the iterations for the components as follows:

- Component C1 - two iterations
- Component C2 - three iterations
- Component C3 - three iterations
- Component C4 - one iteration

These components run differently, depending on whether you group the components or not:

Without Grouping

If you do not group any of the components, the business process test runs each component in sequence: C1 for its iterations, C2 for each of iterations, and so forth:



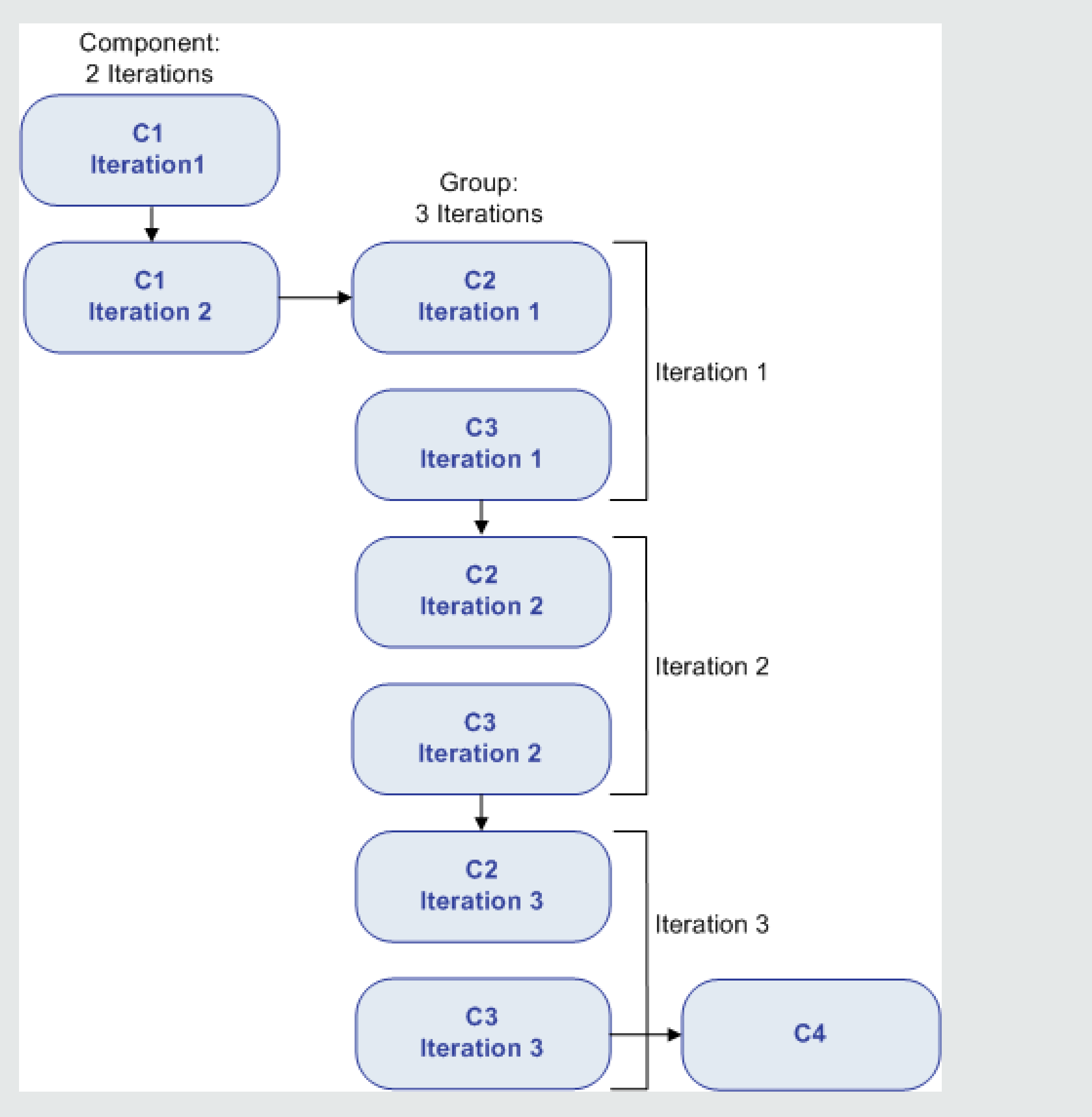
With Grouping

Instead of running all the iterations of component C1, then all the iterations of component C2, and so forth, grouping the components together changes the manner in which the business process test is run.

In this scenario, you group components C2 and C3 together as a group, and set the group to run for three iterations. Thus, the business process test runs in the following order:

- The first iteration of component C1
- The second iteration of component C1
- The first iteration of component C2
- The first iteration of component C3
- The second iteration of component C2
- The second iteration of component C3
- The third iteration of component C2
- The third iteration of component C3
- The single iteration of component C4

This process is illustrated graphically in the example below:



Run Conditions Overview

Relevant for: business process testing

You can use run conditions to insert condition statements into your flows. A **run condition** checks the current value of a component parameter before running the component in a flow. Based on the parameter value and the run condition definition, UFT determines whether to:

- Run the component
- Skip to the next component
- End the component run and set the component status to fail

When you run business process tests containing flows with run conditions, the test run results display the results of run conditions in the test, and lists the components that did not run because a run condition was not met. If a run condition is not met, the test results also provide details about the condition that was not met to help you understand why the component run failed or did not run.

Note: If you set run conditions, and later add or remove a component or change the component order within a flow, the parameters may no longer be relevant and the run condition may not work. For example, if Component B uses an output parameter value from Component A, and you change the order of the components so that Component B precedes Component A, then Component B cannot receive the output parameter value from Component A and the invalid run condition is ignored.

Tasks

How to Set Run Conditions for a Business Process Test

Relevant for: business process testing

The following steps describe how to set run conditions for the flows in a business process test or flow.

> **Note:** This task is part of a higher-level task. For details, see "How to Create, Maintain, and Run Business Process Testing Tests and Flows in UFT" on page 2071.

This task includes the following steps:

- "Prerequisites" below
- "Select the flow or component to set run conditions" below
- "Set On Failure settings for a flow or component" below
- "Set run conditions for a component" on the next page
- "Test the run conditions" on the next page

1. Prerequisites

Ensure that the component for which you are setting run conditions has parameters or that the flow containing the component has a parameter.

2. Select the flow or component to set run conditions

a. Do one of the following:

   - If you are setting run conditions for a flow, in the document pane, select the test containing the flow.

   - If you are setting run conditions for a component, in the document pane, select the business process test or business process flow containing the component.

b. In the document pane, select the flow or component to set run conditions.

3. Set On Failure settings for a flow or component

If you are editing a test, you can set **On Failure** options for all the flows or components included in the test:

a. In the Solution Explorer or document pane, select the business process test you are editing.

b. In the document pane, select the flow or component for which to set On Failure settings.

c. In the Properties pane, open the **Properties** tab.

d. In the Properties tab, set the options for the selected flow or component:

   - **Continue.** Continues the test regardless whether the test passes or fails.
   - **Exits.** Immediately stops the test and exists the test run.

4. Set run conditions for a component

When you are editing a business process flow, you can set additional run options for the components included in the flow:

a. In the Solution Explorer or the document pane, select the business process flow on which you are working.

b. In the document pane, select the component to which you want to add run conditions.

c. In the **Properties** tab in the Properties pane, select the **Use run condition** option.

d. In the middle section of the Properties tab, set the condition options:

   - **Run if:** Indicates what type of parameter (flow or component parameter) and the name of parameter for which a condition must be met
   - **Is:** the operator for the parameter value
   - **Else:** Instructs UFT what to do with the test run if the condition is not met.

5. Test the run conditions

Run the test to ensure that the run conditions were performed correctly.

> **Note:** If a run condition is not valid, the run condition link is displayed in red. This can happen, for example, if a reference parameter was deleted, a parameter value was encrypted, and so on. Delete the run condition and define a new one.

Part 9: Appendix

Appendix A: Where's My UI? (Changes from Previous Versions)

Relevant for: GUI tests and components and API testing

This chapter describes changes that have been made to the user interface between QuickTest Professional 11.00 or Service Test 11.20 and UFT:

Pane Changes

Relevant for: GUI testing and API testing

The table below lists the panes whose names in UFT are different from QuickTest 11.00 orService Test 11.20. The functionality of the pane remains similar.

Previous Pane Name	UFT Pane Name	Additional Information
Available Keywords (QuickTest only)	**Toolbox**	For details on the **GUI testing** (QuickTest) pane, see "Toolbox Pane User Interface (GUI Testing)" on page 510.
Data Table (QuickTest only)	**Data**	For details on the **GUI testing** (QuickTest) Data pane, see "Data Pane Overview" on page 222.
Data Explorer (Service Test only)	**Data**	For details on the **API testing** (UFT) Data pane, see "Data Pane User Interface (API Testing)" on page 252.
Debug Viewer (QuickTest only)	**Debug**	Available **Debug** panes include: • Breakpoints • Call Stack • Watch • Local Variables • Console For details, see " Debug Panes Overview" on page 272.
General (in Application Area) (QuickTest only)	**Properties**	For details, see "General Properties Tab (Properties Pane for Application Areas)" on page 391.
Information (QuickTest only)	**Errors**	The **Information** and **Missing Resources** panes have been merged into a single Errors pane. For details, see "Errors Pane Overview " on page 365.
Missing Resources (QuickTest only)		
Property Sheet (Service Test only)	**Properties**	For details on the Properties pane, see "Properties Pane" on page 385.

Previous Pane Name	UFT Pane Name	Additional Information
Resources (QuickTest only)	**Solution Explorer**	For details on the **Solution Explorer** pane, see "Solution Explorer Pane Overview" on page 471.
Tests (Service Test only)	**Solution Explorer**	
Test Flow (QuickTest only)	**Canvas**	The **canvas** replaces the flow pane and provides all its functionality. For details, see "The Canvas" on page 209.
To Do (QuickTest only)	**Tasks**	For details on the **Tasks** pane, see "Tasks Pane Overview" on page 498.
Toolbox Palette (Service Test only)	**Toolbox**	For details on the **API testing** (UFT) pane, see "Toolbox Pane Overview" on page 506.

Menu Command Changes

Relevant for: GUI testing and API testing

Some of the menu commands from Service Test 11.20 and QuickTest 11.20 are relocated in UFT, to improve product usability. Use this section to locate the commands that moved.

This section discusses the following menu changes:

File Menu Changes

Relevant for: GUI testing and API testing

The following table lists commands from the QuickTest 11.00 or Service Test 11.20 **File** menu and their location in UFT.

Service Test 11.20/ QuickTest 11.00 Menu Option	UFT Option	Additional Information
File > New > Scripted Component (QuickTest only)	**File > New > Business Component**	You can choose all types of business components, including scripted components, through this menu command in the "New <Document> Dialog Box". For details, see the "New <Document> Dialog Box" on page 152.

Service Test 11.20/ QuickTest 11.00 Menu Option	UFT Option	Additional Information
File > Open > Business/Scripted Component (QuickTest only)	**File > Open > Business Component**	You can choose all types of business components through this menu command in the "Open/New <Document>/<Resource> Dialog Box". For details, see the "Open/New <Document>/<Resource> Dialog Box" on page 156.
File > Close All Function Libraries| (QuickTest only)	**Document pane context menu**	Right-click any document tab in the document pane and select **Close All Documents**.
File > ALM/QC Connection	**ALM > (menu option)**	All menu command to manage an ALM connection have been moved to a stand-alone **ALM** menu. **Note:** The ALM Connection is also available by pressing the **ALM Connection** button in the toolbar.
ALM/QC Version Control > Check Out		
ALM/QC Version Control > Undo Check Out		
ALM/QC Version Control > Check In		
ALM/QC Version Control > Version History		
ALM/QC Version Control > Baseline History		
File > Enable Editing (GUI testing only)	Function library context menu	Right-click the tab of a read-only function library in the document pane and select **Enable Editing**.
File > Settings for application areas (GUI testing only)	In Application Area, **Additional Settings** panes	
Process Guidance Management		This command is not relevant for UFT.

Service Test 11.20/ QuickTest 11.00 Menu Option	UFT Option	Additional Information
Associate Function Library <Function Library name> with Test <Test Name>	Document pane context menu	Right-click a function library tab in the document pane and select **Associate Function Library with <Document name>**

Edit Menu Changes

Relevant for: GUI testing and API testing

The following table lists commands from the QuickTest 11.00 orService Test 11.20 **Edit** menu and their location in UFT.

GUI testing (QuickTest) changes

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Edit > Copy Documentation to Clipboard	Keyword View context menu	Right-click on a column header in the Keyword View and select **Copy Documentation to Clipboard**.
Edit > Action > Split Action	This command is not relevant for UFT.	
Edit > Action > Rename Action	Canvas or Keyword View context menu	Right-click an action in the canvas or the Action node in the Keyword View to access all action commands.
Edit > Action > Delete Action		
Edit > Action > Action Properties		
Edit > Action > Action Call Properties		

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Edit > Step Properties > Comment Properties	Keyword View context menu	Right-click the appropriate step in the Keyword View to access the item's properties.
Edit > Step Properties > Object Properties		
Edit > Step Properties > Checkpoint Properties		
Edit > Step Properties > Output Value Properties		
Edit > Step Properties > Report Properties		This option is not available in UFT. Edit the step itself in the Editor or Keyword View.
Edit > Find	**Search > Find** or **Search > Replace**	
Edit > Replace		
Edit > Go To	**Search > Go To**	This menu option also enables you to do multiple types of **Go To** searches.
Edit > Bookmarks	**Search > Bookmarks**	All bookmarks are managed in the Bookmarks pane. For details, see "Bookmarks Overview" on page 204.
Edit > Advanced > Comment Block	**Edit > Format** submenu	These menu commands remain in the **Edit** menu, but are now combined under the **Format** submenu.
Edit > Advanced > Uncomment Block		
Edit > Advanced > Indent		
Edit > Advanced > Outdent		
Advanced > Go To > Function Definition	Editor context menu	Right-click a step in the Editor and select **Go to Function Definition**.

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Advanced > Complete Word	**Edit > Format** submenu	These menu commands remain on the **Edit**, but are now combined under the **Format** submenu.
Advanced > Argument Info		
Advanced > Apply "with" to Script		
Advanced > Remove "With" Statements		
Optional Step		This command is not relevant for UFT.

API testing (Service Test) changes

Service Test 11.20 Menu Option	UFT Option	Additional Information
Select All		This command is not relevant for UFT.
Find and Replace	**Search > Find** **Search > Replace**	
Settings > Show Run Test dialog box		This command is not relevant for UFT.
Settings > Show Results after Run	**Tools > Options > General** tab **> Run Sessions** node	Select the **View results when run session ends** option to enable this command.
Settings > Use Relative Paths in UFT Steps	**Tools > Options > API Testing** tab **> General** node	Select the **Use relative paths to call assets** option to enable this command.
Settings > Activity Repositories	**Tools > Options > API Testing** tab **> General** node	Choose the Activity Repositories path in the **Activity Repositories** area of **General** pane.

View Menu Changes

Relevant for: GUI testing and API testing

The following table lists commands from the QuickTest 11.00 orService Test 11.20 **View** menu and their location in UFT.

GUI testing (QuickTest) changes

QuickTest 11.00 Menu Option	UFT Option	Additional Information
View > Available Keyword	**View > Toolbox**	
View > Data Table	**View > Data**	
View > Debug Viewer	**View > Debug**	This menu command displays a list of debug panes from which to choose.
View > Process Guidance		This command is not relevant for UFT.
View > Resources	**View > Solution Explorer**	
View > Test Flow		This command is not relevant for UFT. The test canvas provides the functionality that was provided by the Test Flow pane.
View > To Do	**View > Tasks**	
View > Keyword View	**View > Keyword View/Editor**	This menu command switches between the Keyword View and Editor. **Note:** You can also change between the Keyword View and the Editor by using the toolbar button.
View > Expert View		
View > Toolbars		This menu command is not available in UFT. For details on the toolbar changes in UFT, see "Toolbar Changes" on page 2158.
Window Theme		This command is not relevant for UFT.

API testing (Service Test) changes

Service Test 11.20 Menu Option	UFT Option	Additional Information
Tests	**View > Solution Explorer**	
Data Window	**View > Data**	

Service Test 11.20 Menu Option	UFT Option	Additional Information
Property Sheet	**View > Properties**	
Window Theme > (suboptions)		This command is not relevant for UFT.

Insert Menu Changes (QuickTest only)

Relevant for: GUI tests and components

The following table lists commands from the QuickTest 11.00 **Insert** menu and their location in UFT.

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Insert > Checkpoint > (suboptions)	**Design > Checkpoint** submenu	All the checkpoints available in QuickTest 11.00 are available under this menu command.
Insert > Output Value > (suboptions)	**Design > Output Value** submenu	All the output values available in QuickTest 11.00 are available under this menu command.
Insert > Step Generator	**Design** submenu	The **Step Generator** command is also available from the context menu for the Keyword View and Editor. Right-click in either view and select **Insert Step > Step Generator**.
Insert > Function Definition Generator		
Insert > Synchronization Point	**Design > Synchronization Point**	
Insert > New Step	Keyword View context menu	Right-click a step in the Keyword View and select **Insert New Step**.
New Step After Block	Keyword View context menu	Right-click a step in the Keyword View and select **Insert New Step After Block**.
Operation		This command is not relevant for UFT
Comment	**Design > Add Comment**	Also available by right-clicking a step in the Keyword View and selecting **Insert Step > Comment**.
Report	Keyword View context menu	Right-click a step in the Keyword View and select **Insert Step > Report**.

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Conditional Statement	**Edit > Code Snippet** submenu	This menu command provides choices of multiple types of conditional and loop statements to insert in a test or component.
Loop Statement		
Call to New Action	**Design > Call to** submenu	Also available by right-clicking an action in the canvas and selecting the relevant commandor a step in the Keyword View/Editor and selecting **Action** and selecting the relevant command.
Call to Copy of Action		
Call to Existing Action		
Call to Service Test test		
Start Transaction	**Design > Start Transaction**	Also available by right-clicking a step in the Editor and selecting **Insert Step > Start Transaction** or **End Transaction**.
End Transaction	**Design > End Transaction**	

Test Menu Changes (Service Test only)

Relevant for: API testing only

The **Test** menu items from Service Test 11.20 are incorporated into other menus in UFT.

Service Test 11.20 Menu Option	UFT Option	Additional Information
Test > Add Reference	**Design > Add Reference**	
Test > Enable Test for Load Testing	**Design > Enable Test for Load Testing**	Only available when HP LoadRunner is installed on the machine.
Test > Open Containing Folder	Context menu in document pane	Right-click on a test tab in the document pane and select **Open Containing Folder in Explorer**.
Test > View Test Results	**View > Last Run Results**	

Automation Menu Changes (QuickTest only)

Relevant for: GUI tests and components

The **Automation** menu from QuickTest 11.00 is incorporated into the **Run** and **Record** menus in UFT.

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Automation > Web Service Testing Wizard	**Tools > Web Service Testing Wizard**	The Web Services Add-in is supported for backwards compatibility only is not enabled by default. New tests and components can use HP's API testing solution for Web service testing purposes. To enable Web Services Add-in for legacy tests, contact HP Software Support.
Automation > Record	**Record > Record**	
Automation > Run	**Run > Run**	
Automation > Stop	**Record > Stop** (if recording) **Run > Stop** (if in a run session)	
Automation > Run Current Action	**Run** submenu	
Automation > Run Maintenance Mode		
Automation > Update Run Mode		
Automation > Analog Recording	**Record > Analog Recording**	

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Automation > Low Level Recording	**Record > Low-Level Recording**	
Automation > Record and Run Settings	**Record > Record And Run Settings**	
Automation > Process Guidance List		This command is not relevant for UFT.
Automation > Results	**View > Last Run Results**	

Build Menu Changes (Service Test only)

Relevant for: API testing only

The **Build** menu from Service Test 11.20 is incorporated into the **Run** menu in UFT.

Service Test 11.20 Menu Option	UFT Option	Additional Information
Build Solution	**Run > Compile submenu**	
Rebuild Solution		
Clean Solution		
Build <test name>		
Rebuild <test name>		
Clean <test name>		
Abort Build		

Service Test 11.20 Menu Option	UFT Option	Additional Information
Set Configuration > Debug	**Tools > Options > API Testing tab > General node**	Select the appropriate option from the drop-down menu for **Compile as** option.
Set Configuration > Release		

Debug Menu Changes

Relevant for: GUI testing and API testing

The **Debug** menu from QuickTest 11.00 orService Test 11.20 is incorporated into the **Run** menu in UFT.

GUI testing (QuickTest) changes

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Pause	**Run** submenu	These commands have all moved into the **Run** menu.
Step Into		
Step Over		
Step Out		
Run to Step		
Debug from Step		
Add to Watch		
Insert/Remove Breakpoint		
Enable/Disable Breakpoint		
Clear All Breakpoints		
Enable/Disable All Breakpoints		

API testing (Service Test) changes

Service Test 11.20 Menu Option	UFT Menu Option
Run Test	**Run** submenu
Stop Process	
Break	
Continue Debugging	
Step Over	
Step Into	
Step Out	
Toggle Breakpoint	

Resources Menu Changes (QuickTest only)

Relevant for: GUI tests and components

The **Associated Function Libraries** command from the QuickTest 11.00 **Resources** menu is not available in UFT.

To view the list of function libraries associated with a test or component, use the "Solution Explorer Pane" (described on page 470), or the "Resources Pane (Test/Business Component Settings Dialog Box) " (described on page 603).

Tools Menu Changes (QuickTest only)

Relevant for: GUI tests and components

The following table lists commands from the QuickTest 11.00 **Tools** menu and their location in UFT.

QuickTest 11.00 Menu Option	UFT Option	Additional Information
View Options	**Tools > Options > Text Editor** tab	Most of the options available in the QuickTest 11.00 View Options are incorporated into the **Text Editor** tab of the Options dialog box in UFT. For details, see "QuickTest Editor Options Dialog Box Changes" on page 2167.
Check Syntax	**Design > Check Syntax**	

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Web Event Recording Configuration	**Record > Web Event Recording Configuration**	
Customize		Not available in UFT.

Window Menu Changes (QuickTest only)

Relevant for: GUI tests and components

The following **Window** menu commands from QuickTest 11.00 are changed in UFT.

QuickTest 11.00 Menu Option	UFT Option	Additional Information
Cascade		These menu commands are not available in UFT. Instead, the functionality of these commands is incorporated into the pane layout. For details, see "UFT Window Layout Customization - Overview" on page 183.
Tile Horizontally		
Tile Vertically.		
Close All Function Libraries	**Window > Close All Documents**	
<windows list>		This menu command is not available in UFT. Instead you can select from the various document tabs available in the document pane.

Help Menu Changes

Relevant for: GUI testing and API testing

The following **Help** menu options from QuickTest 11.00 orService Test 11.20 are changed in UFT.

GUI testing (QuickTest) changes

QuickTest 11.00 Menu Option	UFT Option	Additional Information
QuickTest Professional Tutorial	**Help > UFT Help > Tutorial for GUI Testing**	
Send Feedback and Win		This command is not relevant for UFT.

QuickTest 11.00 Menu Option	UFT Option	Additional Information
HP QuickTest Professional Software Web Page	**Help > Useful Links > Product Page**	

API testing (Service Test) changes

Service Test 11.20 Menu Option	UFT Menu Option	Additional Information
Index		Not available in UFT.
Search		
Contents		
User Guide	**Help > HP UFT Help**	
HP Service Test Web Site	**Help > Useful Links > Product Page**	

Toolbar Changes

Relevant for: GUI testing and API testing

For GUI testing: UFT has a single toolbar, as opposed to the multiple functional toolbars in QuickTest 11.00. The toolbar displays different buttons, depending on the open document or the operations you perform. The functionality of any buttons no longer available in the toolbar is incorporated into menu commands or shortcut (right-click) menu commands.

The sections describe the locations of toolbar buttons or toolbar icons from QuickTest 11.00 orService Test 11.20 that changed.

This section includes:

- "QuickTest Toolbar Buttons" below
- "Service Test Toolbar Buttons" on page 2162

QuickTest Toolbar Buttons

Relevant for: GUI tests and components

The following table shows the new icons for QuickTest 11.00 toolbar buttons, as well as the menu options you can use for buttons that are no longer available.

Standard Toolbar Buttons

QuickTest 11.00 toolbar button		UFT Location	
New Test		**Toolbar button:** New	
Open Test		**Toolbar button:** Open	
Save Test with Resources		**Menu command:** File > Save (Other) > Save with Resources	
Print		Not available in UFT.	
Cut		**Menu commands:** Edit submenu	
Copy			
Paste			
Enable Editing		Function library context menu: Right-click the tab of a read-only function library in the document pane and select **Enable Editing**.	

QuickTest 11.00 toolbar button		**UFT Location**	
Settings		**Menu command:** File > Settings	
ALM/QC Connection		**Toolbar button:** ALM Connection	

Automation Toolbar Buttons

QuickTest 11.00 toolbar button		**UFT Location**	
Web Service Testing Wizard		**Menu command:** Tools > Web Service Testing Wizard **Note:** The Web Services Add-in is supported for backwards compatibility only is not enabled by default. New tests and components can use HP's API testing solution for Web service testing purposes. To enable Web Services Add-in for legacy tests, contact HP Software Support.	
Analog Recording		**On Recording mode dropdown in Record toolbar:** Analog Recording	
Low Level Recording		**On Recording mode dropdown in Record toolbar:** Low-Level	
Standard Windows Recording		**On Recording mode dropdown in Record toolbar** Standard Windows Recording	
Insert Emulator Synchronization Step		**Menu command:** Design > Emulator Synchronization	
Insert Emulator WaitString Step		**Menu command:** Design > Emulator WaitString	
Object Repository		**Toolbar button:** Object Repository	

Debug Toolbar Buttons

QuickTest 11.00 toolbar button		UFT Location	
Pause		**Buttons on Run/Debug Toolbar:** - Pause - Step Into - Step Over - Step Out	
Step Into			
Step Over			
Step Out			
Insert/Remove Breakpoint		**Menu commands:** - Run > Insert/Remove Breakpoint - Run > Clear All Breakpoints - Run > Enable/Disable Breakpoints	
Clear All Breakpoints			
Enable/Disable All Breakpoints			

Edit Toolbar Buttons

QuickTest 11.00 toolbar button		UFT Location
Undo		**Menu command:** Edit > Undo
Redo		**Menu command:** Edit > Redo
Comment Block		**Menu command:** Edit > Format > Comment
Uncomment Block		**Menu command:** Edit > Format > Uncomment
Find		**Menu command:** Search > Find
Replace		**Menu command:** Search > Replace

Insert Toolbar Buttons

QuickTest 11.00 toolbar button		UFT Location
Split Action		This command is not relevant for UFT.
Step Generator		**Menu command:** Design > Step Generator

QuickTest 11.00 toolbar button		UFT Location
Start Transaction		**Menu command:** Design > Start Transaction
End Transaction		**Menu command:** Design > End Transaction

Tools Toolbar Buttons

QuickTest 11.00 toolbar button		UFT Location	
Options		**Menu command:** Tools > Options	
Check Syntax		**Menu command:** Design > Check Syntax	
Object Spy		**Toolbar button:** Object Spy	

View Toolbar Buttons

QuickTest 11.00 toolbar button		UFT Location	
Test Flow		When you open a test in UFT, the test opens in the canvas, which displays the test's flow. This replaces the Test Flow pane.	
Available Keywords		**Toolbar button:** Toolbox	
Resources		**Toolbar button:** Solution Explorer	
Data Table		**Toolbar button:** Data	
Active Screen		**Menu command:** View > Active Screen	
Process Guidance		This option is not available in UFT.	
To Do		**Menu command:** View > Tasks	

Action Toolbar

QuickTest 11.00 toolbar button	UFT Location	Additional Information
<action name>	- Action node in the Solution Explorer - Action on test canvas	
Back	- Test node in the Solution Explorer - Test canvas tab	Select the test name node in the Solution Explorer or the test canvas tab to return to the test flow.
Show	- Action node in the Solution Explorer - Action on test canvas	Double-click the action node in the Solution Explorer or test canvas to open or display the selected action.

Service Test Toolbar Buttons

Relevant for: API testing only

The following table shows the new icons for Service Test 11.20 toolbar buttons, as well as the menu options you can use for buttons that are no longer available.

Service Test 11.20 toolbar button		UFT Location		Additional Information
Open Test	📂	**Toolbar button:** Open	📂 ▾	This toolbar button also provides a drop-down list of document types to open.
Add New Test		**Toolbar button:** Add	+ ▾	This toolbar button also provides a drop-down list of document types (both new and existing to add to a solution).
ALM/QC Connection		**Toolbar button:** ALM Connection	↻	

Enable Test for Load Testing		**Menu command:** Design > Enable Test for Load Testing		Only available when HP LoadRunner is installed on the machine.
Default Zoom	🔍	**Button on canvas**	🔍	
Zoom In	🔍	**Button on canvas**	⊕	
Zoom Out	🔍	**Button on canvas**	⊖	

QuickTest 11.00 Options Dialog Box Changes

Relevant for: GUI tests and components

The UFT Options dialog box (**Tools > Options**) incorporates options from QuickTest 11.00 as well as various new options. The UFT Options dialog box includes tabs for global testing options as well as separate GUI Testing and API Testing tabs. This section lists QuickTest 11.00 options whose location has changed. For details on the Options dialog, see "UFT Global Options" on page 522.

General Pane

The following table lists the **General** pane options that changed from QuickTest 11.00.

QuickTest 11.00 Options String	UFT Path
Display Add-in Manager on startup	**Tools > Options > General** tab **> Startup Options** node
Display Start Page on startup	**Tools > Options > General** tab > **Startup Options** node
Display warning if associated add-ins are not loaded	**Tools > Options > General** tab **> Startup Options** node
Disable recognition of virtual objects while recording	**Tools > Options > GUI Testing** tab **> General** node
Automatically update test and component steps when you rename test objects.	**Tools > Options >** GUI **Testing** tab **> General** node
Display the Keyword View for tests and scripted components	This option is not available in UFT. The Keyword view is always available.
Automatically parameterize steps using...	**Tools > Options >** GUI **Testing** tab **> General** node
Automatically generate "With" statements after recording	**Tools > Options >** GUI **Testing** tab **> General** node
When pointing at a window, activate it after <> tenths of a second	**Tools > Options >** GUI **Testing** tab **> General** node
Restore Layout	**View > Reset Window Layout**
Generate script	**Tools > Options > GUI Testing** tab **> General** node

Run Pane

The following table lists the **Run** pane options that changed from QuickTest 11.00.

QuickTest 11.00 Options String	UFT Path
Run mode	**Tools > Options > GUI Testing** tab **> Test Runs** node
Submit a defect to ALM/QC for each failed step	**Tools > Options > GUI Testing** tab **> Test Runs** node
View results when run session ends	**Tools > Options > General** tab **> Run Sessions** node
Allow other HP products to run tests and components	**Tools > Options > GUI Testing** tab **> Test Runs** node
Stop command shortcut key	This option is not available in UFT. The shortcut key to stop a run is always F4.

Other Options Dialog Panes

The following panes from the QuickTest 11.00 Options dialog have changed locations in UFT:

- **Folders** pane
- **Active Screen** pane
- **Screen Capture** pane
- **Windows Applications** pane
- **Solution Manager** pane
- **Java** pane
- **Terminal Emulator** pane
- **Web** pane
- **SAP** pane
- **Stingray** pane

These panes are found in GUI Testing tab (**Tools > Options > GUITesting** tab).

The options available in these panes have remained in QuickTest 11.00 have remained.

> **Note:** >Some of these panes are displayed only when you load the appropriate add-ins in the "Add-in Manager Dialog Box" (described on page 118) and open a test or component.

Web Services Pane

The Web Services pane from QuickTest 11.00 is not available in UFT.

The Web Services Add-in is supported for backwards compatibility only is not enabled by default. New tests and components can use HP's API testing solution for Web service testing purposes.

To enable Web Services Add-in for legacy tests, contact HP Software Support.

QuickTest Editor Options Dialog Box Changes

Relevant for: GUI tests and components

The UFT Options dialog box also includes the options from the QuickTest 11.00 View Options dialog box. For details on the Options dialog, see "UFT Global Options" on page 522.

The following sections describe the new locations of the QuickTest 11.00 Editor Options dialog:

General Tab

QuickTest 11.00 Editor Options String	UFT Path
Show line numbers	**Tools > Options > Text Editor** tab **> General** node
Auto-indent	This option is not available in UFT. However, the functionality is incorporated into the **Tab spacing** functionality (**Tools > Options > Text Editor** tab **> General** node).
Indent selected text when pressing TAB key	This option is not available in UFT. The Editor is designed to always use the TAB key to indent selected text, if multiple rows are selected.
Statement completion	This option is not available in UFT. The statement completion feature is always enabled.
Draw box around current line	This option is not available in UFT.
Dynamic surround	This option is not available in UFT.
Show all characters	**Tools > Options > Text Editor** tab **> General** node
Auto-expand VBScript syntax	This option is not available in UFT. VBScript syntax automatic completion is always available.
Use tab character	The functionality of this option is included when you select the **Show tabs** options in the Text Editor pane of the Options dialog box (**Tools > Options > Text Editor** tab **> General** node).

Fonts and Colors Pane

The **Fonts and Colors** pane options from QuickTest 11.00 have remained in UFT. These options are found at **Tools > Options > Text Editor** tab **> Font and Colors** node.

For details on this pane, see "Fonts and Colors Pane (Options Dialog Box > Text Editor Tab)" on page 577.

Key Binding Pane

The **Key Binding** pane options from QuickTest 11.00 are not available in UFT.

Appendix B: UFT Terminology Quick Reference

Relevant for: GUI tests and components and API testing

This chapter lists the relevant testing areas for common UFT elements, as well as references for more details.

Testing type	Includes these testing documents:
GUI tests	• GUI tests • actions • function libraries
GUI components	• keyword GUI components • scripted GUI components • application areas • function libraries
API testing	• API tests • API components • user code files

Term (A-Z)	Relevant for:	For details, see:
.NET assembly	API testing	".NET Assembly Activities" on page 1736
accessibility checkpoints	• GUI tests • scripted GUI components	"Accessibility Checkpoints Overview" on page 1400
actions	• GUI tests • API testing	• **For GUI testing:** "Actions in GUI Testing - Overview" on page 872 • **For API testing:** "Actions in API Testing - Overview" on page 1787

Term (A-Z)	Relevant for:	For details, see:
active screen	- GUI tests - scripted GUI components	- "Active Screen Overview" on page 194 - "Active Screen Pane User Interface" on page 200
activity	API testing	"Activity Overview" on page 1612
activity repository	API testing	"How to Perform Activity Sharing" on page 1761
add-ins	- GUI tests - GUI components	*HP Unified Functional Testing Add-ins Guide*
ALM project	- GUI tests - GUI components - API testing	"ALM Integration Overview" on page 709
application area	GUI components	- "Application Areas - Overview" on page 2096 - "Application Area User Interface" on page 2101
array properties	API testing	"Array Control Buttons" on page 438
automation	- GUI tests - GUI components	"UFT Automation Object Model Overview" on page 1156
bitmap checkpoints	- GUI tests - scripted GUI components	"Bitmap Checkpoints Overview" on page 1401
bookmarks	- GUI tests - scripted GUI components - function libraries - API testing	- "Bookmarks Overview" on page 204 - "Bookmarks Pane User Interface" on page 206

Term (A-Z)	Relevant for:	For details, see:
breakpoints	- GUI tests - scripted GUI components - function libraries - API testing	- "Breakpoints " on page 681 - " Breakpoints Pane" on page 277
business process test	- GUI components - API testing	"Business Process Testing in UFT - Overview" on page 2065
canvas	- GUI tests - API testing	"The Canvas" on page 209
checkpoint	- GUI tests - GUI components - API testing	- **For GUI testing:** "Checkpoints Overview" on page 1396 - **For API testing:** "Checkpoint Validation for Test Steps" on page 1612
checkpoint validation	API testing	"Checkpoint Validation for Test Steps" on page 1612
classes	API testing	"How to Search for References or Classes in Documents in the Editor (API Testing Only)" on page 325
code completion	- GUI tests - scripted GUI components - function libraries - API testing	"Automatic Code Completion" on page 316
code object	- GUI tests - GUI components - API testing	

Term (A-Z)	Relevant for:	For details, see:
code snippets	• GUI tests • scripted GUI components	• "Automatic Code Completion" on page 316 • "How to Use Code Snippets and Templates" on page 322
code templates	• GUI tests • scripted GUI components • function libraries • API testing	• "How to Use Code Snippets and Templates" on page 322 • "Code Templates Pane (Options Dialog Box > Coding Tab)" on page 573
compile	API testing	"Output Pane Overview" on page 380
conditional statement	• GUI tests • scripted GUI components	"Comments, Control-Flow, and Other VBScript Statements " on page 1006
conditional steps	API testing`	"Flow Control Activities" on page 1651
data driving	• GUI tests • API testing	• **For GUI testing:** "Data Driver" on page 1543 • **For API testing:** "How to Data Drive an API Test Step" on page 1827
data table	• GUI tests • scripted GUI components • API testing	• **For GUI testing:** "Data Pane Overview" on page 222 • **For API testing:** "Using Data in Your API Test or Component Steps" on page 1801
data table parameters	GUI tests	"Data Table Parameters" on page 1533
data table property	API testing	• "Linking Property Values to a Data Source" on page 1803 • "How to Assign Data to API Test/Component Steps" on page 1819

Term (A-Z)	Relevant for:	For details, see:
database checkpoints	- GUI tests - scripted GUI components	"Database Checkpoints Overview" on page 1405
database query	API testing	"Creating a Query to Retrieve Database Tables" on page 1805
Editor	- GUI tests - scripted GUI components - function libraries - API testing	- "Editing Text and Code Documents" on page 305 - "Editor User Interface" on page 334
environment variables	- GUI tests - GUI components	"Environment Variables" on page 1493
event handler	API testing	"Writing Code for API Test Events - Overview" on page 1938
file content checkpoints	- GUI tests - scripted GUI components	"File Content Checkpoints Overview" on page 1406
function library	- GUI tests - GUI components	"Function Library Overview" on page 1030
global data sheet	- GUI tests - scripted GUI components	"Global Sheet" on page 226
IBM Websphere MQ	API testing	"IBM WebSphere MQ Activities" on page 1699
input/output properties	API testing	"Properties Pane Overview" on page 386

Term (A-Z)	Relevant for:	For details, see:
Insight objects	- GUI tests - GUI components	"Identifying Objects Using Insight" on page 1176
JMS transport	API testing	"JMS Activities" on page 1691
JSON	API testing	- "How to Send a JSON Request to a REST Service" on page 1752 - "How to Receive a JSON Response from a REST Service" on page 1753
Keyword View	- GUI tests - GUI components	- "Keyword View Overview" on page 921 - "Keyword View User Interface" on page 942
load testing	API testing	"Load Testing Activities" on page 1698
log tracking	- GUI tests - GUI components	- "Log Tracking" on page 586 - "Log Tracking Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 617
loop statement	- GUI tests - scripted GUI components	"Comments, Control-Flow, and Other VBScript Statements " on page 1006
maintenance run mode	- GUI tests - GUI components	"Maintenance Run Mode" on page 1080
missing resources	- GUI tests - GUI components - API testing	"Errors Pane Overview " on page 365
multipart HTTP requests	API testing	"How to Send a Multipart HTTP Request" on page 1626
negative testing	API testing	"Negative Testing of Web Services" on page 1738

Term (A-Z)	Relevant for:	For details, see:
object repository	- GUI tests - GUI components	"Object Repository Window - Overview" on page 1260
object spy	- GUI tests - GUI components	"Object Spy Dialog Box " on page 1189
optional steps	- GUI tests - scripted GUI components	"Optional Steps" on page 637
output value	- GUI tests - scripted GUI components	"Output Values Overview" on page 1488
parameter (both input and output)	- GUI tests - GUI components	- "Parameterizing Values Overview " on page 1527 - "Output Values Overview" on page 1488
property	- GUI tests - GUI components - API testing	**For GUI testing:** "Properties Pane User Interface (GUI Testing)" on page 387 **For API testing:** "Properties Pane User Interface (API Testing) " on page 402
recording	- GUI tests - GUI components	"Recording GUI Tests and Components - Overview" on page 843
recovery scenario	- GUI tests - GUI components	"Recovery Scenarios Overview" on page 1112
references	API testing	"Add Reference Dialog Box" on page 485

Term (A-Z)	Relevant for:	For details, see:
regular expression	- GUI tests - scripted GUI components	"Regular Expressions Overview" on page 318
repository parameters	- GUI tests - GUI components	"Shared Object Repositories Overview" on page 1286
resources	- GUI tests - GUI components	"Resources Pane (Test/Business Component Settings Dialog Box) " on page 603
REST Service	API testing	- "REST Service Activities" on page 1735 - "How to a Create a REST Service" on page 1746
run results	- GUI tests - GUI components - API testing	*HP Run Results Viewer User Guide*
run session	- GUI tests - GUI components - API testing	- **For GUI testing:** "Running GUI Tests and Components - Overview " on page 635 - **For API testing:** "Running API Tests - Overview " on page 636
run-time object	- GUI tests - GUI components	"How UFT Applies the Test Object Model Concept" on page 1171
SOAP	API testing	"Web Service Scenario Overview" on page 1879
solution	- GUI tests - GUI components - API testing	"Solutions in UFT Overview" on page 471

Term (A-Z)	Relevant for:	For details, see:
statement completion	- GUI tests - scripted GUI components - function libraries - API testing	"Statement Completion in the Editor" on page 306
step generator	- GUI tests - scripted GUI components	"Step Generator Dialog Box" on page 983
syntax errors	- GUI tests - GUI components - API testing	"Errors Pane Overview " on page 365
system monitor	- GUI tests - GUI components	- "Local System Monitor" on page 585 - "Local System Monitor Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 623
Test Batch Runner	- GUI tests - API testing	"Test Batch Runner Overview" on page 637
test loops	API testing	"Flow Control Activities" on page 1651
test object	- GUI tests - GUI components	"Test Object Model - Overview" on page 1168
test variables	API testing	"How to Define API Test Properties or User/System Variables" on page 141
text checkpoint **text area checkpoints**	- GUI tests - scripted GUI components	"Checking Text Overview " on page 1408

Term (A-Z)	Relevant for:	For details, see:
text recognition	- GUI tests - GUI components	- "Text Recognition Overview" on page 1408 - "Text Recognition Pane (Options Dialog Box > GUI Testing Tab)" on page 542
TODO comments	- GUI tests - scripted GUI components - function libraries - API testing	- "Tasks Pane Overview" on page 498 - "Tasks Pane User Interface" on page 501
Toolbox	- GUI actions - GUI components function libraries - API testing Business process tests and flows	"Toolbox Pane" on page 505
Update Run Mode	- GUI tests - GUI components	"Update Run Mode " on page 1081
Update Service/Assembly	API testing	"Updating Web Services" on page 1854
user code files	API testing	"Writing Code for API Test Events - Overview" on page 1938
Userlogger	API testing	"UserLogger Object" on page 2004
virtual object	- GUI tests - scripted GUI components	"Virtual Objects - Overview" on page 1385

Term (A-Z)	Relevant for:	For details, see:
virtualized services	API testing	"Running Tests with Virtualized Services - Overview" on page 665
Web services	API testing	"Web Service Activities " on page 1734
WSDL	API testing	• "Web Service Activities " on page 1734 • "How to Import a WSDL-Based Web Service" on page 1744
XML checkpoints	• GUI tests • GUI components	"XML Checkpoints Overview" on page 1416
XPath Checkpoints	API testing	"XPath Checkpoints" on page 1613

Appendix C: UFT Installation Utilities

Relevant for: GUI tests and components

This chapter describes the Additional Installation Requirements utility that enables you to configure additional settings in Unified Functional Testing, launch the License Wizard during installation, and install any prerequisite software for working with Unified Functional Testing.

This chapter includes:

Additional Installation Requirements Utility

Relevant for: GUI tests and components

Several prerequisites for working with UFT must be installed and configured after UFT is installed. In addition, to take advantage of UFT debugging and remote access features, certain Internet Explorer and DCOM settings need to be configured.

The Additional Installation Requirements screen displays any prerequisite software that must be installed or configured to work with UFT, according to the options selected when UFT is installed or the installation is modified.

The Additional Installation Requirements utility enables you to configure the required settings automatically, and to run the License Installation Wizard and Microsoft Debugger installation during the UFT installation.

- Select an option name in the Additional Installation Requirements dialog box to display a description of the feature.

- Select the check box of one or more required options and click **Run**.

You can run the Additional Installation Requirements utility at any time by choosing **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Additional Installation Requirements**. In addition to the options to configure Internet Explorer and DCOM settings, and to run the License Wizard, the dialog box displays any prerequisite software that is still required to be installed to work with UFT.

> **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

Other Additional Installation Prerequisites

Additional installation requirements listed in the Additional Installation Requirements dialog box can include the following, according to the options selected when UFT is installed or the installation is modified:

- **Microsoft Script Debugger.** Provides the debugging environment that UFT uses during test runs. This item is displayed only when not currently installed.

- **Terminal Emulator Wizard.** Provides a wizard that enables you to configure terminal emulator identifying settings. This item is displayed only when the Terminal Emulator (TE) Add-in is installed.

- **Stingray Wizard.** Provides a wizard that enables you configure UFT to work with your Stingray application. This item is displayed only when the Stingray Add-in is installed.

  > **Note:** The Web Services Add-in is supported for backwards compatibility only and is not enabled by default. New tests and components can use HP's API testing solution for web

> service testing purposes. To enable the Web Services Add-in for previously created tests, contact HP Software support.

In addition, the Additional Installation Requirements utility enables you to perform the following:

- **Configure Internet Explorer settings.** Select this check box to automatically configure the Internet Explorer options that enable UFT to use the Microsoft Script Debugger application during test runs.

  You can configure these options manually before running UFT. In Internet Explorer, select **Tools > Internet Options > Advanced**. Then select **Disable script debugging** and **Enable third-party browser extensions**.

- **Configure DCOM settings for ALM Integration.** Select this check box to automatically change DCOM permissions and security settings and open a firewall port on your UFT computer. These changes are only required if you want to run UFT tests remotely from ALM, and are running UFT on Windows 7. If you need to set these options manually, see the topic on Modifying DCOM Permissions Manually to Enable Remote Execution in the *HP Unified Functional Testing Installation Guide*. You can also automatically configure DCOM later by running the Additional Installation Requirements tool (**Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Additional Installation Requirements**) or running the Remote Agent (<installation directory>\bin\AQTRmtAgent.exe).

- **Configure DCOM settings for Automation Scripts.** Select this check box to automatically change DCOM permissions and security settings to enable another computer to remotely control UFT using automation scripts.

  > **Caution:** Selecting this option enables remote users to control UFT on this machine, exposing the UFT computer to security risks.

  For information on how to configure this option manually, see the topic on Modifying DCOM Permissions Manually to Enable Remote Execution in the *HP Unified Functional Testing Installation Guide*.

- **Run License Installation Wizard.** Select this check box to run the Unified Functional Testing License Wizard. If you want to run UFT using the demo license for up to 30 days, do not select this check box.

  > **Note:** This option is not automatically selected when upgrading from QuickTest Professional version 9.5 or later because license data is retained. If you did not install a license before the upgrade, select this check box to run the Unified Functional Testing License Wizard.

Appendix D: Understanding Unified Functional Testing Licenses

Relevant for: GUI testing and API testing

This chapter provides a description of the Unified Functional Testing License Installation wizard and the License Validation Utility.

This chapter includes:

Understanding the Unified Functional Testing License Installation Wizard

Relevant for: GUI testing and API testing

The Unified Functional Testing License Installation wizard enables you to install a concurrent or seat license for UFT, or change from one license type to the other. After you select the type of license to install, you follow the steps of the wizard.

This section provides an overview of the Unified Functional Testing License Installation Wizard screens. For more details about the Unified Functional Testing installation and the way the UFT license mechanism works, see the *HP Unified Functional Testing Installation Guide*.

This section describes the following:

Unified Functional Testing License Installation - License Type Screen

Relevant for: GUI testing and API testing

In the License Type screen, you select the type of license with which you want to work. On server operating systems this screen is not displayed and the "Unified Functional Testing License Installation - Welcome Screen" is displayed automatically.

The License Type screen contains the following options:

License Type	Description
Seat license	A permanent license that is specific to the computer on which it is installed. When you first install UFT, this license includes a 30-day demo period during which you must contact HP to obtain a permanent seat license key. After you receive the seat license key, you can activate it to work with UFT permanently. For details, see "Unified Functional Testing License Installation - Welcome Screen" below
Concurrent license	A concurrent license regulates the number of UFT concurrent users. You can work with a concurrent license only if a concurrent license server is installed on your local network, and that license server has at least one available license that is not currently in use. For details, see "Unified Functional Testing License Installation - Concurrent License Server Screen " on page 2188.

Unified Functional Testing License Installation - Welcome Screen

Relevant for: GUI testing and API testing

After selecting the license type (as described in "Unified Functional Testing License Installation - License Type Screen" on the previous page), the Welcome screen opens. The Welcome screen contains the following information:

Information	Description
https://h30580.www3.hp.com/poeticWeb/session/hppHome.htm	A link to the HP Licensing for Software Portal page. Click this link to request a license key.
Information area	The information you need to provide when requesting a seat license. • **Product name.** The name of the product and the version number, for example, Unified Functional Testing. • **Locking code.** A number used to uniquely identify the computer. A locking code from one computer cannot be used on another computer. • **Order number.** The order number printed on the Entitlement Certificate that was shipped with your LTU or ELTU package.

Requesting a License Key via the HP Licensing for Software Portal

The HP Software Licensing Portal assists you in requesting a license key.

For details on the various areas of the portal, see the links under the **Resources** section on the left side of the Web page, such as the tutorial or the how-to demos.

Request your license key as described in the HP Software License Activation Quick Start Guide, available from the **Resources** area of the HP Software Licensing Portal.

Unified Functional Testing License Installation - License Key Screen

Relevant for: GUI testing and API testing

After receiving your license key (described in "Unified Functional Testing License Installation - Welcome Screen" on the previous page), you use the License Key screen to enter the license key you received from HP. The license key is included in the .dat file attached to the email containing your Permanent Password Certificate.

To install the license key:

1. Open the .dat file attached to the email containing your Permanent Password Certificate using any text editor. The license key is also included in the Permanent Password Certificate.

2. Copy the license key (with or without the # character) from the .dat file to the clipboard.

3. Paste the key into the License Installation - License Key screen by clicking the **Paste From Clipboard** button 📋 .

After you click **Next**, the "Unified Functional Testing License Installation - Install Summary Screen" on the next page (described on page 2187) opens.

Notes:

- The license key is valid only for the computer with the locking code that you entered in the license request form.

- A computer with multiple bootable partitions may generate a different locking code for each partition. If a different locking code is generated for a partition, you will need to request a unique license key for it.

Unified Functional Testing License Installation - Install Summary Screen

Relevant for: GUI testing and API testing

After installing your license (as described in "Unified Functional Testing License Installation - License Key Screen" on the previous page), the Install Summary screen displays information about the license you are about to install. The Install Summary screen contains the following information:

Information	Description
License Name	The name of the license you are installing, for example, Unified Functional Testing.
License Type	The type of license you are installing, for example, Seat License.
Expiration Date	The time period for which the license is valid.

After the installation ends, the "Unified Functional Testing License Installation - Finish Screen for Seat Licenses" (described on page 2187) opens.

Unified Functional Testing License Installation - Finish Screen for Seat Licenses

Relevant for: GUI testing and API testing

The Finish screen displays information about the success of the seat license installation. If the license installation is successful, a success message is displayed. If the license could not be installed successfully, a message is displayed describing why. For example, if a seat license was previously installed on your computer, and you try to install a seat license again using the same license key, the license installation will not succeed.

The Finish screen also contains the **Install another license** check box. If you select this check box and click **Next**, the license wizard opens the License Type screen, enabling you to install another license.

Unified Functional Testing License Installation - Concurrent License Server Screen

Relevant for: GUI testing and API testing

After selecting **Concurrent License** in the "Unified Functional Testing License Installation - License Type Screen" on page 2184, you can specify the concurrent license server to which you want to connect. The Concurrent License Server screen has the following options:

Option	Description
Concurrent License Server edit box	Enables you to specify the concurrent license server to which you want to connect. Note that the server must be accessible from the network. You can enter the server's: • host name, for example, MyServer or MyServer.com • IP address, for example, 191.191.19.1 **Note:** • If you do not specify a concurrent license server, UFT will try to locate one for you the next time you open UFT. • If you want to specify the name of a concurrent license server that is currently unavailable, but will be available later, you can enter the name of the concurrent license server in the edit box. Even though the Concurrent License Server screen informs you that the specified server is not available for connection, the next time you open UFT, UFT will try to locate the server you specified.
Check Connection button	Instructs UFT to check that the concurrent license server you specified is accessible from the network. **Note:** If you do not click the **Check Connection** button, UFT checks the connection for you when you click **Next**.

Note:

When you specify one server name:

- The LSFORCEHOST environment variable is created with the server name you specified, instructing UFT to search for available licenses only in the server you specified.
- If you already have an LSHOST environment variable defined, it is replaced with the new LSFORCEHOST environment variable.

When you specify multiple server names or no server name:

- The LSHOST environment variable is created with all of the servers you specified (or none), instructing UFT to search for available licenses first in the server you specified. If no available licenses are found in the specified servers, it tries to locate a server elsewhere on the network.
- If you already have an LSFORCEHOST environment variable defined, it will take precedence over the new LSHOST variable. To enable the LSHOST variable, you must manually delete the LSFORCEHOST environment variable.

  For more details, see "Setting the LSHOST or LSFORCEHOST Variable" on page 2194.

You can specify a different concurrent license server at any time by rerunning this wizard (**Help > License Wizard**), or by modifying the LSHOST or LSFORCEHOST environment variable. For details, see the *HP Unified Functional Testing Installation Guide*.

After the installation ends, the "Unified Functional Testing License Installation - Finish Screen for Concurrent Licenses" opens.

Unified Functional Testing License Installation - Finish Screen for Concurrent Licenses

Relevant for: GUI testing and API testing

The Finish screen displays information about the success of the concurrent license installation. If the license is successful, a success message is displayed. UFT also displays an informative message, if needed. For example, if UFT could not locate a concurrent license server on your network, a message informs you of this.

If you already have an LSFORCEHOST environment variable defined, it will take precedence over the new LSHOST variable. To enable the LSHOST variable, you must manually delete the LSFORCEHOST environment variable.

For more details, see "Setting the LSHOST or LSFORCEHOST Variable" on page 2194.

Validating UFT Licenses

Relevant for: GUI testing and API testing

The License Validation Utility decodes and validates UFT license strings. This enables you to view and copy license information and license validation information for troubleshooting purposes.

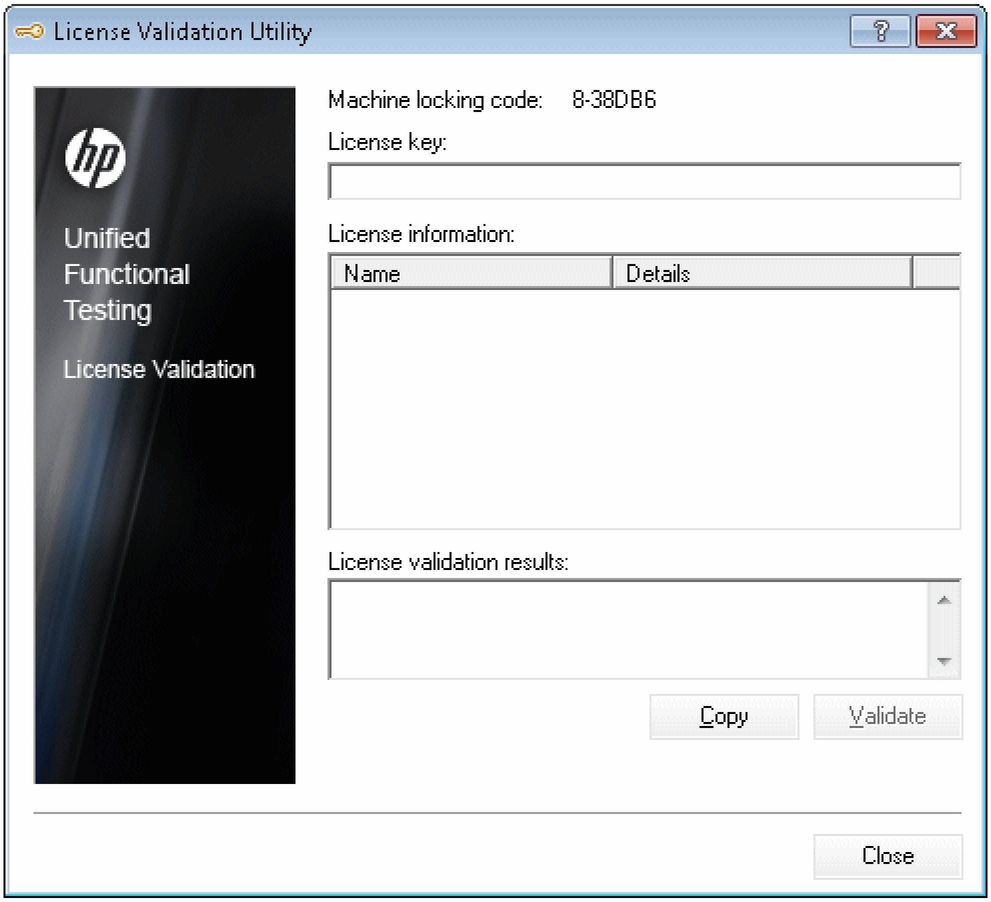
The License Validation Utility performs the following operations:

- Decodes the license string and retrieves important information regarding the license. For details, see "License Information" on page 2192.
- Validates the license according to a predefined set of checks. For details, see "Validation Checks" on page 2193.

If required, you can copy the decoding and validation results to the clipboard. For details, see "Copying Your License Validation Result to the Clipboard" on page 2193.

To decode and validate a license:

1. Select **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > License Validation Utility**.


Machine locking code: 8-38DB6

License Validation Utility
Unified Functional Testing
License Validation
License key:
License information:
Name
Details
License validation results:
Copy
Validate
Close

Note: For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.

2. In the **License Key** box, enter the license code you want to decode and validate. You can find the license code already installed on a UFT computer in the lservrc file. This file is usually located in%APPDATA%\HP\Functional Testing\License.

   Tip: Ensure that a # character is inserted at the end of the license code, or an error will be reported in the **License validation results** area. The # indicates the end of the license code, and any string after the # character is ignored by the License Validation Utility.

3. Click **Validate**. The license string is decoded.

   License information is displayed in the **License information** area. For details, see "License Information" on the next page.

The result of the validation is displayed in the **License validation results** area. For details, see "Validation Checks" on the next page.

4. If required, click **Copy** to copy the information to the clipboard. The copied information includes the current computer locking code, the license string that was decoded, and the decoding and validation results. For details, see "Copying Your License Validation Result to the Clipboard" on the next page.

5. Click **Close** to close the utility.

License Information

Relevant for: GUI testing and API testing

The result of the decoding operation includes the following information about the license.

> **Note:** Some information provided by the operation is intended for Software Support only and is not described here.

- **Feature Name.** The UFT feature name specified when the license was created.
- **Feature Version.** The license version specified when the license was created. This is not the UFT version number.
- **Seat/Concurrent.** The license type. This can be either a **Seat** license that is specific to the computer on which it is installed, or a **Concurrent** license, which references a current license server that can be used by multiple UFT users.
- **Trial/Normal.** The license type. It can be either a **Trial** license, which is a demo license that has a limited period of use, or a **Normal** license.
- **Trial Days Count.** Applicable only for trial licenses. Specifies the number of days until the trial period is over.
- **Locking Code.** The locking code specified when the license was created. This code uniquely identifies the computer on which UFT is installed.
- **Clock Tamper.** Indicates whether a license can be issued, based on whether any date changes have been made to the computer on which UFT is installed.
- **Commuter License.** Indicates whether commuter licenses are supported. A commuter license enables you to work with UFT when you are not connected to the UFT concurrent license server. Commuter licenses are available only with concurrent licenses. For details, see the *HP Unified Functional Testing Installation Guide.*

Validation Checks

Relevant for: GUI testing and API testing

The validation checks performed by the License Validation Utility include the following:

1. Does the UFT feature name match one of the existing features?

2. Does the license version match one of the existing versions?

3. Does the locking code match the locking code of the computer on which UFT is installed?

4. Has the trial period specified in the license string ended?

5. If the license is a concurrent license, does it support commuter licenses?

Copying Your License Validation Result to the Clipboard

Relevant for: GUI testing and API testing

In some cases, you may need a copy of the information provided by this utility. For example, you may need to forward this information to Software Support.

When the validation operation is complete, click the **Copy** button to copy the information to the clipboard. Then paste the information as required.

Setting the LSHOST or LSFORCEHOST Variable

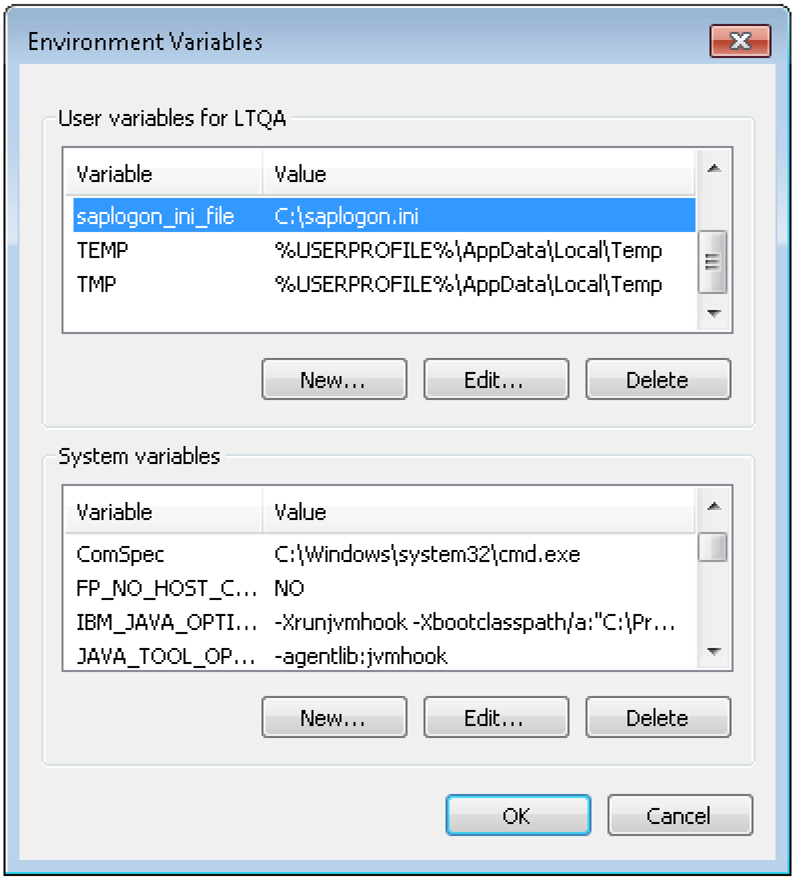
Relevant for: GUI testing and API testing

In addition to using the License Wizard to specify the concurrent license server to which you want UFT to connect, you can also specify a concurrent license server using Windows Environment variables. For example, you can use the LSHOST variable to set the preferred concurrent license server for a UFT client. If the specified concurrent license server cannot be found in the local subnet, then a search is conducted on the entire network. Alternatively, you can use the LSFORCEHOST variable to restrict UFT to a specific concurrent license server.

> **Note:** If you activated a concurrent license using the License Installation Wizard and specified a single server name, the LSFORCEHOST user variable is already defined with the concurrent license server you specified, and cannot be replaced with the LSHOST variable by running the License Installation Wizard again.

To set the LSHOST or LSFORCEHOST variables:

1. Open the Environment Variables dialog box. (Select **Control Panel > System > Advanced system settings > Environmental Variables** button (for Windows 7 or Vista users) or **Control Panel > System > Advanced** tab > **Environmental Variables** button (for Windows XP users). The Environment Variables dialog box opens.)


Environment Variables

User variables for LTQA

Variable	Value
saplogon_ini_file	C:\saplogon.ini
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
IBM_JAVA_OPTI...	-Xrunjvmhook -Xbootclasspath/a:"C:\Pr...
JAVA_TOOL_OP...	-agentlib:jvmhook

> **Note:** If the concurrent license server is already defined using an LSHOST or LSFORCEHOST system variable, you do not need to define it using the user variable.

2. Under **User variables for <user name>**, click **New**. The New User Variable dialog box opens.

3. In the **Variable Name** box, type LSHOST or LSFORCEHOST.

4. In the **Variable Value** box, type the full name of the concurrent license server. Alternatively, you can enter the IP address of the host computer.

> **Note:**
>
> - The LSHOST and LSHOSTFORCE environment variable values are limited to 64 characters.
>
> - For multiple host names (for example, where there are HP Functional Testing Concurrent License Servers running on one network), use a semicolon (;) to separate the host names or IP addresses in the **Variable Value** box.

5. Click **OK** to close the New User Variable dialog box.

6. Click **OK** to close the Environment Variables dialog box.

7. Click **OK** to close the System Properties dialog box.

To delete the LSFORCEHOST variable:

1. In the User variables list of the Windows Environment Variables dialog box, select the LSFORCEHOST variable.

2. Click **Delete**, and restart UFT for the changes to take effect.

Troubleshooting and Limitations - UFT Licenses

Relevant for: GUI testing and API testing

This section describes troubleshooting and limitations for UFT licenses.

- As documented in the *HP Unified Functional Testing Installation Guide*, seat licenses are not supported on server operating systems. In some cases, the license installation wizard may not disable the option to select **Seat** license on the operating system. If you do select it, the license key installation will fail.

- When you connect to a Vista or Windows XP 64-bit operating system using a Remote Desktop Connection, you cannot select to install a seat license or modify the UFT license type from **Concurrent** to **Seat**.

  Workaround: Open UFT on the physical computer to change the license type, or connect to the computer using a console that does not create a session when it accesses the remote computer.

- The HP Functional Testing Concurrent License Server does not support the use of Network Address Translation (NAT).

- You cannot install UFT with a demo license over a QuickTest Professional installation whose demo license has expired and for which a regular license was never installed.

- If the LSHOST variable was set to a server on another domain, the server utility lsmon.exe might behave unexpectedly.

- The concurrent license does not include a demo license and does not work without an HP Functional Testing Concurrent License Server and an installed license key.

- You cannot change the license type from seat to concurrent or vice versa when logged into the machine without administrator privileges.

  Workaround: To change the license type, log into the machine as a user with administrator privileges.

- If version 6.0.0.8169 of Pdm.dll is found on your computer, the setup program will recognize this during the UFT installation and will instruct you to download the corrected DLL from the Microsoft site. For more details, see http://support.microsoft.com/kb/q293623/.

- Installing UFT on a network drive is not supported.

- When working with a terminal server (for example, Windows 2003 server), you must connect to UFT using a concurrent license. Seat and demo licenses are not supported on terminal servers.

  Workaround: Install a concurrent license server on the terminal server and then connect to it, or connect to a concurrent license server that is installed on a different computer.

Chapter E: Developing Custom Comparers for Bitmap Checkpoints (GUI Testing)

Relevant for: GUI tests and components

This appendix is intended for COM programmers who want to customize the algorithm used to compare bitmaps in bitmap checkpoints.

This chapter includes:

Concepts

About Developing Custom Comparers for Bitmap Checkpoints

Relevant for: GUI tests and components

A custom comparer is a COM object that you develop to run the bitmap comparison in a bitmap checkpoint according to a specific algorithm. This enables you to create bitmap checkpoints that perform the comparison according to your needs.

By default, a bitmap checkpoint in UFT compares the actual and expected bitmaps pixel by pixel and fails if there are any differences. UFT provides various bitmap checkpoint configuration options that enable you to refine the bitmap comparison and make it more flexible. For example, you can define tolerance levels, or you can instruct UFT not to compare complete images, but rather compare selected areas within them, or to locate a specific image within an object in your application. For more details, see "Bitmap Checkpoints Overview" on page 1401.

If you need to further customize the way bitmaps are compared in checkpoints, you can develop custom comparers and install and register them on the UFT computer. A UFT user can then choose to use a custom comparer to perform the comparison in a bitmap checkpoint (on a per checkpoint basis).

The COM object that you develop must implement interfaces that UFT provides in a type library, and register to the component category that UFT defines for bitmap comparers. The type library (BitmapComparer.tlb) and the category ID (defined in ComponentCategory.h) are available in <UFT installation folder>\dat\BitmapCPCustomization.

When a UFT user creates or edits a bitmap checkpoint, UFT displays any registered custom comparers in the advanced settings in the Bitmap Checkpoint Properties dialog box (in addition to the UFT default comparer). The user can then select a comparer according to the testing requirements of the specific application or bitmap being tested. For more details about using custom comparers in UFT, see "Custom Comparers" on page 1404.

You can find an example of a situation where developing a custom comparer enhanced the use of bitmap checkpoints, in "Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario" below.

Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario

Relevant for: GUI tests and components

Ben is a quality assurance engineer who is experienced in using UFT, and often uses bitmap checkpoints to test the appearance of different icons or pictures in the user interface he is testing. He does not have a programming background.

Joanne is a software engineer who is experienced in image processing and is familiar with COM programming.

When Ben began testing the user interface of a furniture purchasing application, he created a bitmap checkpoint to test that the pictures of the items on sale were displayed properly. In the checkpoint, he captured an image of the pane in the application that contained the pictures he wanted to test. Ben found that the bitmap checkpoint often failed, even though the graphic images displayed in the application during the run seemed identical to the ones he had captured when creating the checkpoint.

Ben reviewed the actual, expected, and difference bitmaps displayed in the run results. He also took a closer look at the application's user interface. The application contained three panes. The left pane displayed general information, the middle pane displayed the pictures of the items on sale, and the right pane displayed the corresponding list of items and relevant details. Ben found that depending on the information displayed in the left pane, the images in the middle pane sometimes shifted slightly one way or the other within the pane. While the images themselves were still identical, their changed location was causing the bitmap checkpoint to fail.

Ben did not want to use pixel tolerance to address this issue because he wanted the checkpoint to fail when the pixels within the images themselves were not identical.

When Ben mentioned his problem to a co-worker, she suggested that developing a custom comparer for his bitmap checkpoints could solve the problem, and referred him to Joanne. Joanne developed a custom comparer that would accept as input the number of pixels that the images should be allowed to shift without failing the checkpoint. The bitmap comparison that Joanne designed would pass the checkpoint only if the images were identical and they had all shifted by the same number of pixels. This way, Ben knew that his checkpoint would still catch incorrect images and cases where the application's interface looked bad because the images were no longer aligned.

Ben installed and registered the custom comparer on his UFT computer, and then selected the new custom comparer for his bitmap checkpoint. After some experimenting, he found the optimal number of pixels to enter in the configuration string, so that significant changes in the application's interface were detected, but insignificant shifting of the images did not cause the checkpoint to fail.

After Ben successfully used this custom comparer for a while, his company decided to install and register it on all of the UFT computers. The custom comparer would now be available to everyone in the quality assurance team to use for similar situations.

Custom Bitmap Comparer Development

Relevant for: GUI tests and components

To develop a custom comparer, you create a COM object that implements the UFT bitmap checkpoint comparer interfaces (described on page 2220) to perform the following:

- Accept input from UFT and perform the bitmap comparison.
- Provide comparison results to UFT.
- (Optionally) Provide information that UFT can display in the advanced settings in the Bitmap Checkpoint Properties dialog box when a user creates or edits a bitmap checkpoint.

Custom comparers run within the UFT context. You must therefore exercise care when developing your custom comparer, as its behavior and performance will affect the behavior and performance of UFT.

For UFT to recognize the custom comparer, it must be registered to the component category that UFT defines for bitmap comparers. Depending on how you implement your custom comparer, you can design the comparer to register itself when it is installed, or you can provide an additional program that needs to be run at the time of installation. For details, see "How to Install Your Custom Comparer and Register it to UFT" on page 2206.

Perform the tutorial in "How to Develop a Custom Comparer - Tutorial" on page 2211 to learn how to create and use a custom comparer. You can then create your own custom comparers in much the same way. For task details, see "How to Develop a Custom Comparer" on the next page.

In addition to the tutorial, UFT provides source files that implement a sample custom comparer in different languages. The source files are provided in C++ and in Visual Basic. Both projects generate a similar custom comparer.

You can study the samples to help you learn about developing custom comparers for UFT bitmap checkpoints, or use them as a reference or template when you develop your own custom comparers. For details, see "How to Use the Bitmap Checkpoint Custom Comparer Samples" on page 2209.

Tasks

How to Develop a Custom Comparer

Relevant for: GUI tests and components

This task describes the process for developing a custom bitmap comparer.

> **Tip:** To practice performing this task, see "How to Develop a Custom Comparer - Tutorial" on page 2211.

This task includes the following steps:

- "Prerequisites" on page 2209
- "Develop the custom comparer COM object" below
- "Prepare the custom comparer installation - optional " below
- "Install the custom comparer" on the next page
- "Test the custom comparer" on the next page

1. Prerequisites

- Knowledge of image processing
- Experience in developing COM objects

2. Develop the custom comparer COM object

a. Create the custom comparer COM object. You can use any language and development environment that supports creating COM objects.

> **Note:** Depending on the language that you use for development, you might be able to specify the custom comparer name when you create the COM object. Otherwise the name can be specified on the UFT computer after registering the object. For details, see "Set the Custom Comparer Name - Optional" on page 2208.

b. Program your COM object to implement the custom comparer interfaces. For details, see "How to Implement the Bitmap Comparer Interfaces" on page 2204.

3. Prepare the custom comparer installation - optional

Your custom comparer might need to be installed on more than one computer. You can create a program that automatically performs the steps necessary to install and register the comparer and its documentation on those computers.

For details on the steps that such a program needs to perform, see "How to Install Your Custom Comparer and Register it to UFT" on page 2206.

For example, when you design your custom comparer installation, you must ensure that when it is installed on the UFT computer, it is also registered to the component category for UFT bitmap comparers. This can be achieved in different ways, such as:

- If you develop your custom comparer in C++ using Microsoft Visual Studio, you can modify the **DllRegisterServer** and **DllUnregisterServer** methods to handle this registration. These methods are called when you run a .dll using the regsvr32.exe program. You can see an example of this type of implementation in "How to Develop a Custom Comparer - Tutorial" on page 2211.

- If you develop your custom comparer in an environment that does not enable you to modify the registration methods, you can add an additional program that handles this registration and instruct users who install the custom comparer to run this program as well. You can see an example of this type of implementation in the Visual Basic sample custom comparer that UFT provides. For more details, see "How to Use the Bitmap Checkpoint Custom Comparer Samples" on page 2209.

4. **Install the custom comparer**

   On the computer where you want to use the custom comparer, do one of the following:

   - Run the installation program that automatically installs and registers the comparer.

   - Manually install and register the custom comparer. For details, see "How to Install Your Custom Comparer and Register it to UFT" on page 2206.

5. **Test the custom comparer**

   Create bitmap checkpoint steps in a test or component in UFT. In the Bitmap Checkpoint Properties dialog box, click **Advanced settings**, select your custom comparer, and use it to perform bitmap checkpoints and check that they perform correctly.

   > **Tip:** By default, UFT displays expected, actual, and difference bitmaps in the Run Results only for checkpoints that fail. When you test your custom comparer on UFT, you might want to see the expected, actual, and difference bitmaps in the run results even for bitmap checkpoints that pass. To configure this, select **Tools > Options > GUI Testing** tab **> Screen Capture** node in UFT and set the **Save still image captures to results** option to **Always**. For more details on using this option, see "Screen Capture Pane (Options Dialog Box > GUI Testing Tab)" on page 557.

How to Implement the Bitmap Comparer Interfaces

Relevant for: GUI tests and components

This task describes how to implement the bitmap comparer interfaces so that your custom comparer COM object performs the following:

- Accepts bitmaps and compares them
- Provides comparison results to UFT
- Provides information for the advanced settings in the Bitmap Checkpoint Properties dialog box

> **Note:** This task is part of a higher-level task. For details, see "How to Develop a Custom Comparer" on page 2202.

This task includes the following steps:

- "Prerequisite - Reference the type library" below
- "Implement the CompareBitmaps method to accept input and compare bitmaps" below
- "Implement the CompareBitmaps method to return the comparison results to UFT" on the next page
- "Implement IBitmapCompareConfiguration to provide information for the Bitmap Checkpoint Properties dialog box" on the next page

Prerequisite - Reference the type library

In the COM object that you develop, reference the type library that UFT provides (located in <UFT installation folder>\dat\BitmapCPCustomization\BitmapComparer.tlb)

Implement the CompareBitmaps method to accept input and compare bitmaps

UFT calls the **"The CompareBitmaps Method" on page 2220** method in the **IVerifyBitmap** interface (described on page 2220) to pass the expected and actual bitmaps to the custom comparer for comparison.

Method syntax:

```
HRESULT CompareBitmaps    ([in] IPictureDisp* pExpected,
                  [in] IPictureDisp* pActual,
                  [in] BSTR bstrConfiguration,
                  [out] BSTR* pbstrLog,
                  [out] IPictureDisp** ppDiff,
                  [out, retval] VARIANT_BOOL* pbMatch);
```

Implement the **"The CompareBitmaps Method" on page 2220** method to perform the following:

- Accept and compare two bitmaps according to a predefined algorithm that you define based on the testing requirements.
- Accept a text string that can contain configuration information provided by the UFT user (in the Bitmap Checkpoint Properties dialog box, Advanced settings), and use it in the comparison. For example, the string could contain tolerance specifications, acceptable deviations in size or location of the image, or any other information that you want to affect the comparison.

The string can have any format you choose (XML, comma separated, INI file style, and so on). Make sure that the documentation you provide for the custom comparer describes the format. The configuration input that the UFT user enters in the advanced settings in the advanced settings in the Bitmap Checkpoint Properties dialog box must conform to this format.

Implement the CompareBitmaps method to return the comparison results to UFT

UFT displays the results of bitmap checkpoints in the Run Results Viewer.

When you implement the **"The CompareBitmaps Method" on page 2220** method in the **IVerifyBitmap** interface (described on page 2220) to compare the bitmaps, you must also return the following information:

- Whether the bitmaps match and the checkpoint should pass.
- A text string that UFT displays in the run results.

  The purpose of this string is to provide information about the comparison to the UFT user, but while you develop and test your comparer, you can use this string for debugging purposes as well.
- A bitmap that visually represents the difference between the actual and expected bitmaps.

  The purpose of this bitmap is to help the UFT user understand why the checkpoint failed. The custom comparer can create this bitmap using any visualization approach you choose. For example, the default UFT comparer creates a black-and-white bitmap containing a black pixel for every pixel that is different in the two images.

Implement IBitmapCompareConfiguration to provide information for the Bitmap Checkpoint Properties dialog box

When a UFT user selects a custom comparer in the advanced settings of the Bitmap Checkpoint Properties dialog box, UFT displays an **input** text box, and, optionally, a link to documentation provided for the custom comparer. For details, see "Custom Comparers" on page 1404.

To support these options, you can implement the **IBitmapCompareConfiguration** interface (described on page 2222) to provide the necessary information for the dialog box.

- Implement the **"The GetDefaultConfigurationString Method" on page 2222** method to return the default configuration string for your custom comparer.

Method syntax:

```
HRESULT GetDefaultConfigurationString ([out, retval] BSTR* pbstrConfiguration);
```

UFT displays this string in the **Input** box in the advanced settings in the Bitmap Checkpoint Properties dialog box.

The format of this string must be the same as the format of the configuration string that the comparer expects as input.

- Implement the **GetHelpFilename** method to return a path to the documentation about your custom comparer. A UFT user can then access the documentation from the advanced settings in the Bitmap Checkpoint Properties dialog box.

  Method syntax:

  ```
  HRESULT GetHelpFilename ([out, retval] BSTR* pbstrFilename);
  ```

  The documentation can be in any format that you choose. UFT opens the documentation using the program associated with the provided file type on the user's computer. Therefore, you should provide the documentation in a format for which you expect the UFT user to have the necessary program.

  The documentation should provide the UFT user with the following information:

  - The type of comparison the custom comparer performs (to enable the user to determine when to use it to run a bitmap checkpoint).
  - The required format for the configuration string and the possible values it can contain.
  - An explanation of the comparison result information that is displayed in the run results (text string and difference bitmap).

How to Install Your Custom Comparer and Register it to UFT

Relevant for: GUI tests and components

The custom comparer must be installed and registered on any computer that runs a test or component with a bitmap checkpoint using the custom comparer.

This task describes how to install the custom comparer on a UFT computer and register it to UFT.

- When you develop a custom comparer, you can create a program that automatically performs the steps in this task. If you choose not to create an installation program, review these steps and make sure to provide your users with all of the required files and information.

- When you install a custom comparer that you did not develop, you may need additional information from the developer to perform the steps in this task. Alternatively, you may receive an installation program from the developer that performs this task automatically.

> **Note:** This task is part of a higher-level task. For details, see "How to Develop a Custom Comparer" on page 2202.

This task includes the following steps:

- "Prerequisites" below
- "Install the custom comparer COM object on the UFT computer" below
- "Register the custom comparer to the component category for UFT bitmap comparers " below
- "Place the custom comparer documentation in the correct location" on the next page
- "Set the Custom Comparer Name - Optional" on the next page
- "Results" on the next page

Prerequisites

1. More than one custom comparer can be installed and registered on the same UFT computer.

   However, before installing and registering a new version of a specific custom comparer, unregister the existing comparer.

2. A custom comparer .dll is created using a specific development environment version. Make sure that the computer on which this .dll runs has the corresponding runtime environment installed.

Install the custom comparer COM object on the UFT computer

For example, you can do this by double-clicking the .dll or running the .dll using the regsvr32.exe program.

Register the custom comparer to the component category for UFT bitmap comparers

Register the component category ID for UFT bitmap comparers, **CATID_QTPBitmapComparers**, as a registry key under the COM object's **HKEY_CLASSES_ROOT\CLSID\<Object's CLSID>\Implemented Categories** key.

> **Note:** When UFT is installed, it adds this component category ID as a registry key under the **HKEY_CLASSES_ROOT\Component Categories** key. The component category ID is defined in <UFT installation folder>\dat\BitmapCPCustomization\ComponentCategory.h.

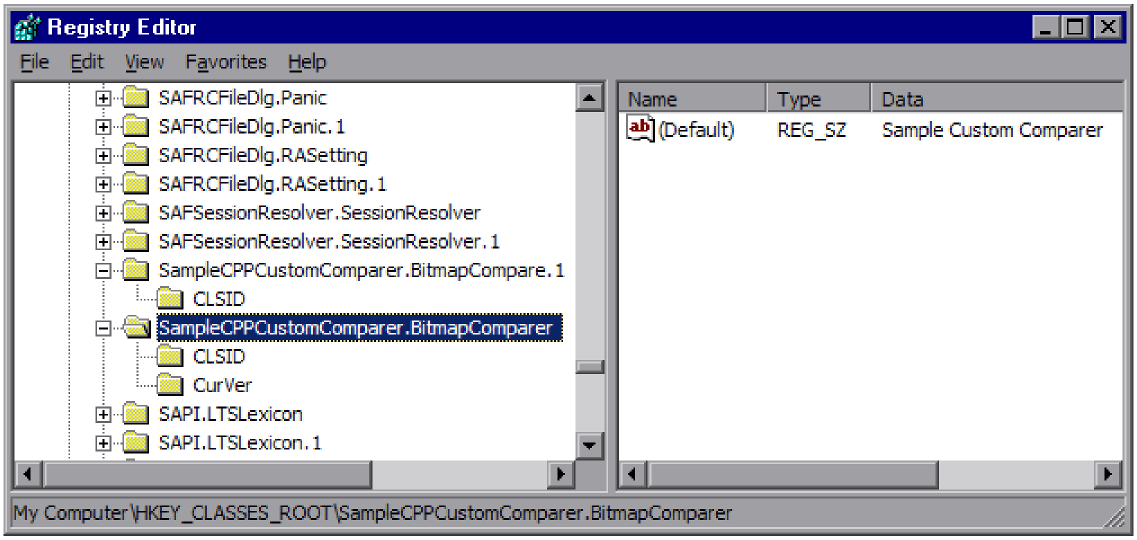
Place the custom comparer documentation in the correct location

The Bitmap Checkpoint Properties dialog box in UFT can display documentation about the custom comparer, if such documentation is provided.

Place the custom comparer documentation in the location specified in the custom comparer object's **GetHelpFilename** method.

Set the Custom Comparer Name - Optional

UFT displays the name of the custom comparer in the advanced settings in the Bitmap Checkpoint Properties dialog box and in the Run Results Viewer. The name that UFT uses is the value (in the registry) of the default property of the custom comparer ProgID key under the HKEY_CLASSES_ROOT key. For example, in the image below, the name of the custom comparer is **Sample Custom Comparer**.



Name	Type	Data
(Default)	REG_SZ	Sample Custom Comparer

My Computer\HKEY_CLASSES_ROOT\SampleCPPCustomComparer.BitmapComparer

- In some environments you set the name while developing the object. For example if the custom comparer is developed in C++ using Microsoft Visual Studio, this name can be specified during development in the **Type** box in the ATL Simple Object Wizard.

- In other environments, you can set or customize the name as part of the installation or registration process on each computer. For example, if the custom comparer is developed in Visual Basic, this registry value is automatically set to the COM object's ProgID. If you want to modify the custom comparer name, you can edit it manually in the registry after the comparer is installed, or design the program that performs the installation and registration to edit this value as well.

Results

In the advanced settings in the Bitmap Checkpoint Properties dialog box, UFT displays the comparer you installed and registered, along with all of the available custom comparers, and the UFT default comparer. You can then select the appropriate comparer to use for each bitmap checkpoint.

How to Use the Bitmap Checkpoint Custom Comparer Samples

Relevant for: GUI tests and components

This task describes how to generate the sample custom comparer, and then register it and work with it.

This task includes the following steps:

- "Prerequisites" below
- "Generate the sample comparer" below
- "Install the custom comparer on a UFT computer" below
- "Register the custom comparer to the component category for UFT bitmap comparers" below
- "Study the custom comparer functionality" on the next page

1. **Prerequisites**

   Decide whether you want to use the sample C++ project or the Visual Basic one.

   The samples are located under <UFT installation folder>\samples\BitmapCPSample.

2. **Generate the sample comparer**

   a. To open the sample project, do one of the following:

      - To open the C++ project, use Microsoft Visual Studio 2003 or later.
      - To open the Visual Basic project, use Microsoft Visual Studio 6.0.

   b. Compile the custom comparer, to build the .dll.

   > **Note:** If you are using Microsoft Visual Studio 2010 to compile the C++ project, make the following modification before compiling: Open the stdafx.h file in <UFT installation folder>\samples\BitmapCPSample\CPPCustomComparer, locate the line #define _WIN32_WINNT 0x0400 and change it to #define _WIN32_WINNT 0x0501.

3. **Install the custom comparer on a UFT computer**

   Run the custom comparer using the regsvr32.exe program to install it on the computer.

4. **Register the custom comparer to the component category for UFT bitmap comparers**

- **If you are using the C++ sample project:**

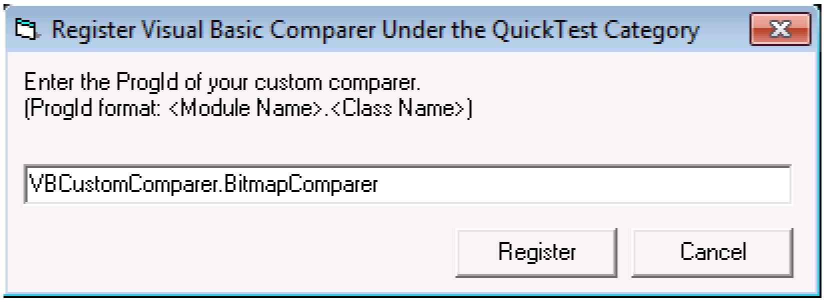
  The C++ sample sources implement registering the custom comparer to UFT in the **DllRegisterServer** and **DllUnregisterServer** methods. Therefore, if you used the C++ project to create the .dll, running the .dll (in the previous step) will also register the custom comparer.

  > **Note:** The name displayed for the custom comparer in UFT will be **Custom QTP Bitmap Comparer**.

- **If you are using the Visual Basic sample project:**

  The Visual Basic sample project does not implement this registration. Therefore, the Visual Basic sample also includes an additional tool that you must run after installing the custom comparer, to register the custom comparer to the component category for UFT bitmap comparers. For more details, see "How to Install Your Custom Comparer and Register it to UFT" on page 2206.

  You can run the Visual Basic Comparer Registration Tool from <UFT installation folder>\samples\BitmapCPSample\VBCustomComparer\RegisterCategory.exe.

  Register Visual Basic Comparer Under the QuickTest Category

  Enter the ProgId of your custom comparer.
  (ProgId format: <Module Name>.<Class Name>)

  VBCustomComparer.BitmapComparer

  In the dialog box that opens, enter the ProgID of the custom comparer and click **Register**.

  > **Note:** The name displayed for the custom comparer in UFT will be its ProgId—**VBCustomComparer.BitmapComparer**. For details on modifying this name, see "Set the Custom Comparer Name - Optional" on page 2208.

5. **Study the custom comparer functionality**

   Create bitmap checkpoint steps in a test or component in UFT. In the advanced settings in the Bitmap Checkpoint Properties dialog box, you can select the sample custom comparer and use it to perform bitmap checkpoints.

   a. Note the customized information that is displayed in the dialog box when you select the custom comparer.

  - The default configuration string that the sample comparer returns (and UFT displays in the Bitmap Checkpoint Properties dialog box) is MaxSurfAreaDiff=140000.

  - The documentation provided with this sample comparer (and opened from the Bitmap Checkpoint Properties dialog box) is the SampleComparerDetails.txt text file located in <UFT installation folder>\samples\BitmapCPSample\CPPCustomComparer.

b. Run bitmap checkpoints to test the behavior of the sample comparer. For example, you can run checkpoints on the Windows Calculator application, alternately setting the Calculator view to **Standard** or **Scientific**, to obtain different size bitmaps for the same object.

   The sample custom comparer does not compare the content of the actual and expected bitmaps. It compares the total number of pixels they contain. For configuration input, this comparer expects a string that defines the MaxSurfAreaDiff parameter. The comparer fails the checkpoint if the difference in total number of pixels is greater than the number defined for MaxSurfAreaDiff.

c. View the results of your checkpoint in the run results.

   This sample bitmap custom comparer returns the actual bitmap as the difference bitmap. In addition, it provides a text string that specifies the difference in total number of pixels. UFT displays this string in the run results.

How to Develop a Custom Comparer - Tutorial

Relevant for: GUI tests and components

This tutorial walks you step-by-step through the process of creating a custom comparer in C++ using Microsoft Visual Studio. The custom comparer you create is similar to the sample custom comparer provided with UFT. You can create your own custom comparers in a similar way. For details about the sample custom comparer, see "How to Use the Bitmap Checkpoint Custom Comparer Samples" on page 2209.

> **Note:** For a task related to this tutorial, see "How to Develop a Custom Comparer" on page 2202.

By following the instructions in this section, you create a COM object that:

- Implements the **CompareBitmaps** method to receive two bitmaps to compare and a configuration string, compare the (size of) the two bitmaps, and return the necessary results.

- Implements the **GetDefaultConfigurationString** method and the **GetHelpFilename** method, to return the information that UFT displays in the advanced settings in the Bitmap Checkpoint Properties dialog box.

- Registers to the component category for UFT bitmap comparers.

When the design of your custom comparer is complete, you can install and register it and use it in UFT to run a bitmap checkpoint.

> **Note:** Depending on the version of Microsoft Visual Studio that you use to perform the tutorial, the command names may be different.

This tutorial includes the following steps:

- "Create a new ATL project—SampleCPPCustomComparer" below
- "Create a new class—CBitmapComparer" below
- "Define that the CBitmapComparer class implements the bitmap checkpoint comparer interfaces" on the next page
- "Move the function bodies for the bitmap checkpoint comparer interface methods from BitmapComparer.h to BitmapComparer.cpp" on the next page
- "Implement the bitmap checkpoint comparer interface methods to customize the bitmap checkpoint as required" on page 2215
- "Design your custom comparer to register to the component category for UFT bitmap comparers" on page 2217
- "Compile your DLL and run it using the regsvr32.exe program" on page 2218
- "Test your custom comparer by using it for bitmap checkpoints in UFT" on page 2218

1. Create a new ATL project—SampleCPPCustomComparer

a. In Microsoft Visual Studio, select **New > Project**. The New Project dialog box opens.

b. Select the **ATL Project** template, enter SampleCPPCustomComparer in the **Name** box for the project, and click **OK**. The New ATL Project wizard opens.

c. In **Application Settings**, make sure that the **Attributed** option is not selected, and click **Finish**.

2. Create a new class—CBitmapComparer

a. In the class view, select the **SampleCPPCustomComparer** project, right-click, and select **Add > Class**. The Add Class dialog box opens.

b. Select **ATL Simple Object** and click **Add**. The ATL Simple Object Wizard opens.

c. In the **Short name** box, enter BitmapComparer. The wizard uses this name to create the names of the class, the interface, and the files that it creates.

d. In the **Type** box, enter Sample Custom Comparer. This is the custom comparer name that UFT will display in the advanced settings in the Bitmap Checkpoint Properties dialog box and in the run results. For details, see "Set the Custom Comparer Name - Optional" on page 2208.

e. Click **Finish**. The wizard creates the necessary files for the class that you added, including .cpp and .h files with implementation of CBitmapComparer class.

3. Define that the CBitmapComparer class implements the bitmap checkpoint comparer interfaces

a. In the class view, select **CBitmapComparer**, right-click, and select **Add > Implement Interface**. The Implement Interface wizard opens.

b. In the **Implement interface from** option, select **File**. Browse to or enter the location of the UFT bitmap checkpoint comparer type library. The type library is located in: <UFT installation folder>\dat\BitmapCPCustomization\BitmapComparer.tlb.

The wizard displays the interfaces available in the selected type library, **IBitmapCompareConfiguration** and **IVerifyBitmap**.

c. Add both interfaces to the list of interfaces to implement, and click **Finish**.

In the BitmapComparer.h file, the wizard adds the declarations, classes, and method stubs that are necessary to implement the interfaces. In subsequent steps you will need to add implementation to these method stubs.

> **Note:** In Microsoft Visual Studio 2005, the wizard generates the signature for the **CompareBitmaps** method in the **IVerifyBitmap** interface incorrectly. To enable your project to compile correctly, manually change the type of the last argument (pbMatch) from BOOL* to VARIANT_BOOL*.

4. Move the function bodies for the bitmap checkpoint comparer interface methods from BitmapComparer.h to BitmapComparer.cpp

a. Open the BitmapComparer.h and BitmapComparer.cpp files.

b. In BitmapComparer.h, create declarations for the bitmap checkpoint comparer interface methods (based on the function bodies that the wizard created): **CompareBitmaps**, **GetDefaultConfigurationString**, and **GetHelpFilename**.

c. Move the function bodies that the wizard created for the bitmap checkpoint comparer interface methods from the BitmapComparer.h file to the BitmapComparer.cpp file.

At the end of this step, BitmapComparer.cpp and BitmapComparer.h should contain the following code:

```
// BitmapComparer.cpp : Implementation of CBitmapComparer
#include "stdafx.h"
#include "BitmapComparer.h"
// CBitmapComparer
// IBitmapCompareConfiguration Methods
STDMETHODIMP CBitmapComparer::GetDefaultConfigurationString
                              (BSTR * pbstrConfiguration)
{
          return E_NOTIMPL;
}
STDMETHODIMP CBitmapComparer::GetHelpFilename(BSTR * pbstrFilename)
{
          return E_NOTIMPL;
}
// IVerifyBitmap Methods
STDMETHODIMP CBitmapComparer::CompareBitmaps
                  (IPictureDisp * pExpected, IPictureDisp * pActual,
                  BSTR bstrConfiguration, BSTR * pbstrLog,
                  IPictureDisp * * ppDiff, VARIANT_BOOL * pbMatch)
{
          return E_NOTIMPL;
}
```

```
// BitmapComparer.h : Declaration of the CBitmapComparer
#pragma once
#include "resource.h"      // main symbols
#include "SampleCPPCustomComparer.h"
// CBitmapComparer
class ATL_NO_VTABLE CBitmapComparer :
      public CComObjectRootEx<CComSingleThreadModel>,
      public CComCoClass<CBitmapComparer, &CLSID_BitmapComparer>,
      public IDispatchImpl<IBitmapComparer, &IID_IBitmapComparer,
               &LIBID_SampleCustomComparerLib, /*wMajor =*/ 1, /*wMinor =*/ 0>,
      public IDispatchImpl<IBitmapCompareConfiguration,
               &__uuidof(IBitmapCompareConfiguration),
               &LIBID_BitmapComparerLib, /* wMajor = */ 1, /*wMinor =*/ 0>,
      public IDispatchImpl<IVerifyBitmap, &__uuidof(IVerifyBitmap),
               &LIBID_BitmapComparerLib, /* wMajor = */ 1, /*wMinor =*/ 0>
{
   public:
      CBitmapComparer()
      {
      }
      DECLARE_REGISTRY_RESOURCEID(IDR_BITMAPCOMPARER)
      BEGIN_COM_MAP(CBitmapComparer)
        COM_INTERFACE_ENTRY(IBitmapComparer)
        COM_INTERFACE_ENTRY2(IDispatch, IBitmapCompareConfiguration)
        COM_INTERFACE_ENTRY(IBitmapCompareConfiguration)
        COM_INTERFACE_ENTRY(IVerifyBitmap)
      END_COM_MAP()
      DECLARE_PROTECT_FINAL_CONSTRUCT()
      HRESULT FinalConstruct()
      {
        return S_OK;
      }
      void FinalRelease() {}
      // IBitmapCompareConfiguration Methods
public:
      STDMETHOD(GetDefaultConfigurationString)(BSTR * pbstrConfiguration);
      STDMETHOD(GetHelpFilename)(BSTR * pbstrFilename);
      // IVerifyBitmap Methods
public:
      STDMETHOD(CompareBitmaps)(IPictureDisp * pExpected,
            IPictureDisp * pActual, BSTR bstrConfiguration, BSTR * pbstrLog,
               IPictureDisp * * ppDiff, VARIANT_BOOL * pbMatch);
};
OBJECT_ENTRY_AUTO(__uuidof(BitmapComparer), CBitmapComparer)
```

5. **Implement the bitmap checkpoint comparer interface methods to customize the bitmap checkpoint as required**

   In this tutorial, you implement a custom comparer similar to the sample custom comparer provided with UFT. For details about the sample custom comparer, see "How to Use the Bitmap Checkpoint Custom Comparer Samples" on page 2209.

   When you create your own custom comparers, this is the step during which you design the custom comparer logic. You define the configuration input that it can receive, the algorithm that it uses to compare the bitmaps, and the output that it provides.

   In the BitmapComparer.cpp file, add #include <atlstr.h>, and implement the bitmap checkpoint comparer interface methods as follows:

   - The **GetDefaultConfigurationString** method:

```
STDMETHODIMP CBitmapComparer::GetDefaultConfigurationString
                    (BSTR * pbstrConfiguration)
{
    CComBSTR bsConfig("MaxSurfAreaDiff=140000");
    *pbstrConfiguration = bsConfig.Detach();
    return S_OK;
}
```

- The **GetHelpFilename** method:

```
STDMETHODIMP CBitmapComparer::GetHelpFilename(BSTR * pbstrFilename)

{
    CComBSTR bsFilename ("..\\samples\\BitmapCPSample\\
   CPPCustomComparer\\SampleComparerDetails.txt");
    *pbstrFilename = bsFilename.Detach();
    return S_OK;
}
```

> **Note:** When the GetHelpFilename method returns a relative path, UFT searches for this path relative to <UFT installation folder>\bin. The implementation above instructs UFT to use the documentation file provided with the CPP sample custom comparer.

- The **CompareBitmaps** method:

```
STDMETHODIMP CBitmapComparer::CompareBitmaps
                  (IPictureDisp * pExpected, IPictureDisp * pActual,
                  BSTR bstrConfiguration, BSTR * pbstrLog,
                  IPictureDisp * * ppDiff, VARIANT_BOOL * pbMatch)
{
    HRESULT hr = S_OK;
    if (!pExpected || !pActual)
      return S_FALSE;
    CComQIPtr<IPicture> picExp(pExpected);
    CComQIPtr<IPicture> picAct(pActual);
    // Try to get HBITMAP from IPicture
    HBITMAP HbmpExp, HbmpAct;
    hr = picExp->get_Handle((OLE_HANDLE*)&HbmpExp);
    if (FAILED(hr))
      return hr;
    hr = picAct->get_Handle((OLE_HANDLE*)&HbmpAct);
    if (FAILED(hr))
      return hr;
    BITMAP ExpBmp = {0};
    if( !GetObject(HbmpExp, sizeof(ExpBmp), &ExpBmp) )
```

```
            return E_FAIL;
        BITMAP ActBmp = {0};
        if( !GetObject(HbmpAct, sizeof(ActBmp), &ActBmp) )
            return E_FAIL;
        CString s, tol;
        tol = bstrConfiguration;
        int EPos = tol.ReverseFind('=);
        tol = tol.Right(tol.GetLength() - EPos - 1);
        int maxSurfaceAreaDiff = _ttoi(tol);
        // Set output parameters
        CComPtr<IPictureDisp> Diff(pActual);
        *ppDiff = Diff;
        int DiffPixelsNumber = abs (ExpBmp.bmHeight * ExpBmp.bmWidth - ActBmp.bmHei
ght * ActBmp.bmWidth);
        *pbMatch = DiffPixelsNumber <= maxSurfaceAreaDiff;
        s.Format(_T("The number of different pixels is: %d."), DiffPixelsNumber);
        CComBSTR bs (s);
        *pbstrLog = bs.Detach();
        return hr;
}
```

6. Design your custom comparer to register to the component category for UFT bitmap comparers

For UFT to recognize the COM object that you create as a custom comparer, you must register it to the component category for UFT bitmap comparers. The component category ID is defined in <UFT installation folder>\dat\BitmapCPCustomization\ComponentCategory.h.

You can implement this registration in the **DllRegisterServer** and **DllUnregisterServer** methods in the SampleCPPCustomComparer.cpp file that the wizard created as part of your project. These methods are called when you run a DLL using the regsvr32.exe program.

a. Add the <UFT installation folder>\dat\BitmapCPCustomization folder to your project's include path.

b. Open the SampleCPPCustomComparer.cpp file and add the following line: #include "ComponentCategory.h"

c. In the SampleCPPCustomComparer.cpp file, modify the **DllRegisterServer** and **DllUnregisterServer** methods created by the wizard, to contain the following code:

```
STDAPI DllRegisterServer(void)
{
        // registers object, typelib and all interfaces in typelib
```

```
     HRESULT hr = _AtlModule.DllRegisterServer();
    CComPtr<ICatRegister> spReg;
    hr = spReg.CoCreateInstance
        (CLSID_StdComponentCategoriesMgr, 0, CLSCTX_INPROC);
    if (FAILED(hr))
      return hr;
    // register comparer to the UFT bitmap comparers category
    CATID catid = CATID_QTPBitmapComparers;
    hr = spReg->RegisterClassImplCategories(CLSID_BitmapComparer, 1, &catid);
    return hr;
}

STDAPI DllUnregisterServer(void)
{
    HRESULT hr = _AtlModule.DllUnregisterServer();
    CComPtr<ICatRegister> spReg;
    hr = spReg.CoCreateInstance
        (CLSID_StdComponentCategoriesMgr, 0, CLSCTX_INPROC);
    if (FAILED(hr))
      return hr;
    // unregister comparer from the UFT bitmap comparers category
    CATID catid = CATID_QTPBitmapComparers;
    hr = spReg->UnRegisterClassImplCategories(CLSID_BitmapComparer, 1, &catid);
    return hr;
}
```

Note the second section in these methods, that handles registration to the component category for UFT bitmap comparers—**CATID_QTPBitmapComparers**.

7. **Compile your DLL and run it using the regsvr32.exe program**

   Your custom comparer can now be used in UFT for bitmap checkpoints.

8. **Test your custom comparer by using it for bitmap checkpoints in UFT**

   For details on how to work with bitmap checkpoints, see "Bitmap Checkpoints Overview" on page 1401.

   a. Open UFT and create a bitmap checkpoint on the Windows Calculator application (Standard view).

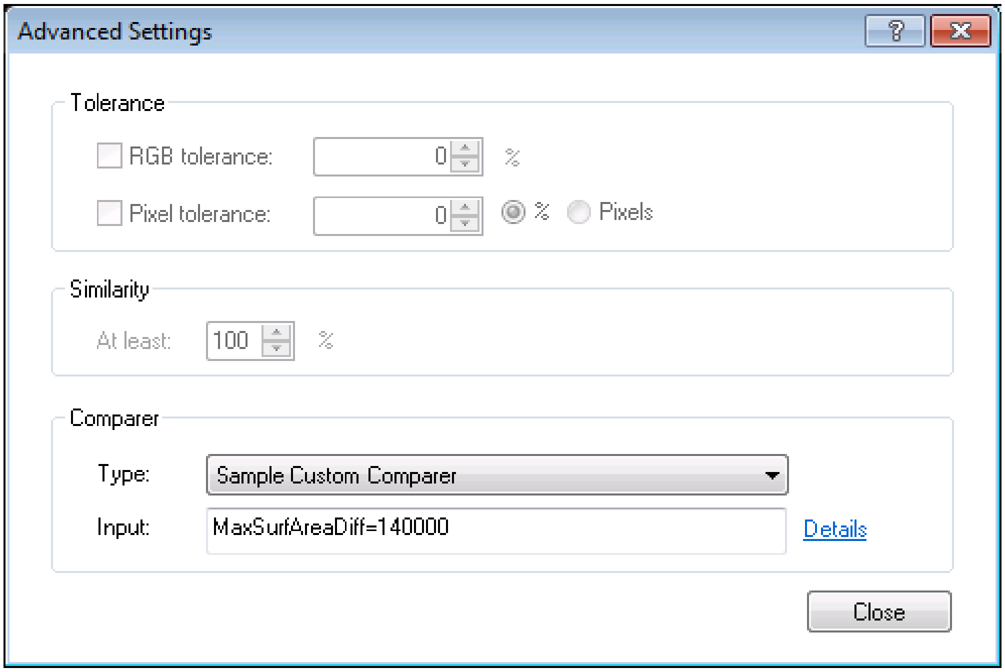
      The advanced settings in the Bitmap Checkpoint Properties dialog box include the **Comparer** option, in which you can select the default UFT comparer or your sample custom comparer.

   b. Change the Calculator view to **Scientific**. The size of the calculator object is now larger. Run the checkpoint using the default UFT comparer. The checkpoint fails.

c. Edit the checkpoint in the Bitmap Checkpoint Properties dialog box:

   - Make sure the **Compare selection with runtime bitmap** checkpoint mode is selected.

   - Click **Advanced settings** to open the Advanced Settings dialog box, and select **Sample Custom Comparer** in the Comparer **Type** box.



Tolerance

□ RGB tolerance: 0 %

□ Pixel tolerance: 0 ● % ○ Pixels

Similarity

At least: 100 %

Comparer

Type: Sample Custom Comparer

Input: MaxSurfAreaDiff=140000

In the **Input** box, you can see the default configuration string returned by the **GetDefaultConfigurationString** method: MaxSurfAreaDiff=140000

In the **Input** box, you can see the default configuration string returned by the GetDefaultConfigurationString method: MaxSurfAreaDiff=140000

If you click **Details**, the text file containing documentation for the sample custom comparer opens.

The comparer you designed in this exercise checks how different the expected and actual bitmaps are in size, and fails the checkpoint if the difference is greater than the number of pixels defined in the configuration string.

If you run the checkpoint using default MaxSurfAreaDiff value, the checkpoint passes, because the difference in the total size of the calculator object when it is set to different views is less than 140000 pixels (the difference is approximately 80000 pixels). If you set MaxSurfAreaDiff to 70000, the checkpoint fails.

View the run results to see the text string and difference bitmap that your custom comparer provides to UFT after the comparison.

Reference

The Bitmap Checkpoint Comparer Interfaces

Relevant for: GUI tests and components

Your custom comparer must implement the interfaces described in this section. UFT calls these interfaces' methods when creating or running a bitmap checkpoint that uses your custom comparer.

This section includes:

The IVerifyBitmap Interface

Relevant for: GUI tests and components

This interface contains the CompareBitmaps method that you need to implement to perform the bitmap comparison for the checkpoint.

The CompareBitmaps Method

The **CompareBitmaps** method receives the actual and expected bitmaps that need to be compared for the bitmap checkpoint, and a string that can contain configuration input for the custom comparer.

The method must compare the bitmaps according to the comparison algorithm for which this custom comparer is designed, and return the results to UFT.

The results include:

- An indication whether the bitmaps match and the checkpoint should pass.
- A text string that contains information about the results of the bitmap comparison.
- A bitmap that reflects the differences between the actual and expected bitmaps.

UFT displays the results that this method returns in the Run Results Viewer. For details, see the section on Bitmap Checkpoint Results (described in the *HP Run Results Viewer User Guide*).

Method syntax:

```
HRESULT CompareBitmaps ([in] IPictureDisp* pExpected,
                        [in] IPictureDisp* pActual,
                        [in] BSTR bstrConfiguration,
                        [out] BSTR* pbstrLog,
                        [out] IPictureDisp** ppDiff,
                    [out, retval] VARIANT_BOOL* pbMatch);
```

Method Parameters:

- *pExpected.* A picture object (input).

  The expected bitmap stored in the checkpoint.

- *pActual.* A picture object (input).

  The actual bitmap captured from the application being tested.

- *bstrConfiguration.* A text string (input).

  A string that contains configuration input for the custom comparer. This is the string displayed in the **Input** box in the advanced settings in the Bitmap Checkpoint Properties dialog box.

  The string can be the default configuration string that the custom comparer provides to UFT in the **GetDefaultConfigurationString** method described below, or an input string entered by the UFT user.

  The **bstrConfiguration** string can have any format you choose (XML, comma separated, ini file style, and so on). Make sure that the default configuration string returned by the **GetDefaultConfigurationString** method matches the format expected in the **CompareBitmaps** method. Additionally, make sure that the documentation you provide for your custom comparer explains the format that the UFT user must use when editing this string in the **Input** box.

- *pbstrLog.* A text string (output).

  A string that contains information about the results of the bitmap comparison. UFT displays this string in the Run Results Viewer.

- *ppDiff.* A picture object (output).

  A bitmap (created by the custom comparer) that reflects the difference between the actual and expected bitmaps. UFT displays this bitmap in the Run Results Viewer along with the actual and expected bitmaps.

- *pbMatch.* A boolean value (output).

  A value that indicates whether the bitmaps match and the checkpoint should pass.

  Possible values:

  - VARIANT_TRUE. Actual and expected bitmaps match, checkpoint passes.

  - VARIANT_FALSE. Actual and expected bitmaps do not match, checkpoint fails.

Return Value

The HRESULT that this method returns indicates whether the comparison ran successfully (and not whether the bitmaps match).

The IBitmapCompareConfiguration Interface

Relevant for: GUI tests and components

This interface contains the methods that you need to implement to support the custom comparer options that UFT displays in the advanced settings in the Bitmap Checkpoint Properties dialog box. For details, see "Checkpoint Properties Dialog Box" on page 1432.

The GetDefaultConfigurationString Method

The **GetDefaultConfigurationString** method must return the default configuration string for your custom comparer. For details on configuration strings, see "Implement the CompareBitmaps method to accept input and compare bitmaps" on page 2204.

UFT displays this string in the **Input** box in the advanced settings in the Bitmap Checkpoint Properties dialog box when a user creating a new bitmap checkpoint selects your custom comparer.

If the UFT user does not modify the input string in the dialog box, the string provided by **GetDefaultConfigurationString** is passed to the custom comparer's **CompareBitmaps** method. You must therefore make sure that the default configuration string matches the format that your custom comparer expects to receive in the **CompareBitmaps** method.

Method syntax:

```
HRESULT GetDefaultConfigurationString ([out, retval] BSTR* pbstrConfiguration);
```

The GetHelpFilename Method

The **GetHelpFilename** method must return a path to the documentation that contains information about your custom comparer for UFT users.

UFT displays the documentation when a user selects your custom comparer in the advanced settings in the Bitmap Checkpoint Properties dialog box and clicks **Details**. Make sure that when your custom comparer is installed, the documentation that you provide is installed in the location specified by the **GetHelpFilename** method.

The path can be one of the following:

- A full path to a file.
- A relative path to a file (UFT searches for this path relative to <UFT installation folder>\bin).
- A URL.

If you do not provide documentation for your custom comparer, this method should return the HRESULT E_NOTIMPL. For details on the type of information you should provide, see "Implement

IBitmapCompareConfiguration to provide information for the Bitmap Checkpoint Properties dialog box" on page 2205.

Method syntax:

```
HRESULT GetHelpFilename ([out, retval] BSTR* pbstrFilename);
```

Chapter F: GUI Checkpoints and Output Values Per Add-in

Relevant for: GUI tests and components

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

Relevant for: GUI tests and components

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 page 2227.

	Accessibility	**Bitmap**	**Database**	**File Content**	**Image**	**Page**	**Standard**	**Table**	**Text**	**Text Area**	**XML (Application)**	**XML (Resource)**
.NET Web Forms[3]	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
Oracle	NA	S	NA	NA	NA	NA	S	S	NS	NS	NA	NA

	Accessibility	Bitmap	Database	File Content	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
PeopleSoft	S	S	NA	NA	S	S	S	S	S[1]	NS	S	S
PowerBuilder[2]	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
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

	Accessibility	**Bitmap**	**Database**	**File Content**	**Image**	**Page**	**Standard**	**Table**	**Text**	**Text Area**	**XML (Application)**	**XML (Resource)**
Web	S	S	NA	NA	S	S	S	S	S[1]	S	S[7]	NA
Web Services	NA	NA	NA	NA	NA	NA	S	NA	NA	NA	S	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.

Note: For additional information about using text recognition in checkpoints, see "Guidelines for Text Recognition" on page 1409.

Supported Output Values

Relevant for: GUI tests and components

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 page 2230.

	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
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

	Accessibility	Bitmap	Database	File Content	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
PowerBuilder[2]	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
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

	Accessibility	Bitmap	Database	File Content	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
Web Services	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	S
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.

Appendix G: Using Performance Testing and Business Service Management Products with UFT GUI Tests

Relevant for: GUI tests only

This chapter includes:

Concepts

HP Performance Testing and Business Service Management Products Overview

Relevant for: GUI tests only

UFT enables you to create complex tests that examine the full spectrum of your application's functionality to confirm that every element of your application works as expected in all situations.

After you use UFT to create and run a suite of tests that test the functional capabilities of your application, you may want to test how much load your application can handle or to monitor your application as it runs.

- **HP performance testing products (LoadRunner and Performance Center)** test the performance and reliability of an entire system under controlled and peak load conditions. To generate load, these performance testing products run hundreds or thousands of virtual users. These virtual users provide consistent, repeatable, and measurable load to exercise your application just as real users would.

- **HP Business Service Management (formerly HP Business Availability Center)** enables real-time monitoring of the end user experience. Business Process Monitor runs synthetic users to perform typical activities on the monitored application.

If you have already created and perfected a test in UFT that is a good representation of your user’s actions, you may be able to use your test as the basis for performance testing and application management activities.

You can use **Silent Test Runner** to check in advance that a test will run correctly from LoadRunner, Performance Center, and Business Process Monitor.

Features for Use with Performance Testing and Business Service Management

UFT offers several features that are designed specifically for integration with LoadRunner, Performance Center, and Business Process Monitor.

> **Note:** These products are designed to run tests using virtual or synthetic users representing many users simultaneously performing standard user operations. Some features may not be available when integrating these products with UFT.

You can use the **Services** object and its associated methods to insert statements that are specifically relevant to Performance Testing and Business Service Management. These include:

AddWastedTime	EndTransaction	SetTransaction
EndDistributedTransaction	LogMessage	SetTransactionStatus

GetEnvironmentAttribute	Rendezvous	ThinkTime
StartDistributedTransaction	StartTransaction	UserDataPointUserDataPoint

For details on these methods, see the **Services** section of the *HP UFT Object Model Reference for GUI Testing* and your HP performance testing or Business Service Management documentation.

For details on transactions, see "Measuring Transactions" on page 2237.

For details on inserting StartTransaction and EndTransaction statements using the UFT menu options, see "Start Transaction Dialog Box" on page 2243 and "End Transaction Dialog Box" on page 2242.

Advantages of Running Tests in LoadRunner

The main advantages of running tests in LoadRunner are:

- To check how your application's functionality is affected by heavy load
- To measure the response time that a typical user experiences on the client side while your application is under load (end-to-end response time)

For example, you can add tests to specific points in a LoadRunner scenario to confirm that the application's functionality is not affected by the extra load at those sensitive points.

Another advantage of using a virtual GUI user (GUI Vuser) script as part of your LoadRunner scenario is that the GUI Vuser script runs on your screen during the scenario, enabling you to watch the actual steps executed by the Vuser in real time.

Designing Tests for Use HP Performance Testing and HP Business Service Management Products

If you plan to use the same test in both UFT and LoadRunner, Performance Center, and/or Business Process Monitor, you should take into account the different options supported in each product as you design your test. For details, see:

- "Designing Tests for HP Performance Testing Products" on the next page
- "Designing GUI Tests for HP Business Process Monitor" on page 2235

Running Tests from HP Performance Testing and HP Business Service Management Products

The run mechanisms used in all HP Performance Testing and HP Business Service Management products are the same. This means that you can create tests that are compatible with LoadRunner, Performance Center, and Business Process Monitor, enabling you to take advantage of tests or test segments that have already been designed and debugged in UFT.

For example, you can add tests to specific points in a performance test to confirm that the application's functionality is not affected by the extra load at those sensitive points. You can also run tests on Business Process Monitor to simulate end user experience and ensure that your application is running correctly and in a timely manner.

If you plan to use the same test in both UFT and LoadRunner, Performance Center, and/or Business Process Monitor, you should take into account the different options supported in each product before you run your test. For details, see:

- "Running GUI Tests from HP Performance Testing Products " below
- "Running GUI Tests from HP Business Process Monitor" on page 2236

Designing Tests for HP Performance Testing Products

Relevant for: GUI tests only

Consider the following guidelines when designing tests for use with performance testing products:

- The tests you use with LoadRunner and Performance Center should be simple, designed to pinpoint specific operations, and should avoid using external actions and references to other external files (including resources stored in ALM). Also, when working with action iterations, corresponding StartTransaction and EndTransaction statements must be contained within the same action.
- Every test must contain at least one transaction to provide useful information in the performance test. LoadRunner and Performance Center use only the data that is included within a transaction, and ignore any data in a test outside of a transaction.
- Do not include references to external actions or other external resources (including resources stored in ALM), such as an external data table file, environment variable file, shared object repositories, function libraries, and so forth. This is because LoadRunner or Performance Center may not have access to the external action or resource.

  (However, if the resource can be found on the network, UFT will use it. For example, you can try defining external resources via an absolute path, or by adding them as supplementary files and transferring them to Load Generator in the GUI test folder.)
- Make sure that the last step(s) in the test closes the application being tested, as well as any child processes that are running. This enables the next iteration of the test to open the application again.

For details on working with LoadRunner or Performance Center, see your HP performance testing documentation.

Running GUI Tests from HP Performance Testing Products

Relevant for: GUI tests only

Consider the following guidelines when running tests from HP Performance Testing Products:

- You can run only one GUI Vuser concurrently per computer. (A GUI Vuser is a Vuser that runs a GUI test.)

- Ensure that UFT is closed on the UFT computer before running a test in Performance Center or LoadRunner.

- The settings in the LoadRunner or Performance Center Run-time Settings dialog box are not relevant for tests.

- You cannot use the **ResultDir** environment variable when running a performance test.

- Transaction breakdown is not supported for tests (scripts) created with UFT.

- UFT cannot run on a computer that is:

  - Logged off or locked. In these cases, consider running UFT on a terminal server.

  - Already running a test. Make sure that the test is finished before starting to run another test.

  > **Tip:** You can simulate how the test will run from a performance testing product by using Silent Test Runner. For details, see "Silent Test Runner" on page 2239.

Designing GUI Tests for HP Business Process Monitor

Relevant for: GUI tests only

> **Note:** As of Business Service Management 9.0, profiles are no longer used. Instead, Business Process Monitor uses Business Transaction Flows, which are included in the parent application's run unit and can be run as part of that unit or independently, as needed.

Consider the following guidelines when designing tests for use with Business Process Monitor:

- The tests you use with Business Process Monitor should be simple, designed to pinpoint specific operations, and should avoid using external actions and references to other external files (including resources stored in ALM). Also, when working with action iterations, corresponding **StartTransaction** and **EndTransaction** statements must be contained within the same action.

- Every test must contain at least one transaction to provide useful information in Business Process Monitor. Business Process Monitor uses only the data that is included within a transaction, and ignores any data in a test outside of a transaction.

- Business Process Monitor does not support running tests that require access to external resources, including resources stored in ALM (such as a shared object repository, function library, external data table, external actions, and so forth). Tests that require external resources may fail to run on Business Process Monitor. (However, if the resource can be found on the network, UFT will use it.)

- Make sure that the last steps in the test close the application being tested, as well as any child processes that are running. This cleanup step enables the next test run to open the application again.

- When measuring a distributed transaction over two different Business Process Monitor profiles or Business Transaction Flows (depending on the version), the profile with the **StartDistributedTransaction** statement must be run before the profile with the associated **EndDistributedTransaction**.

- When measuring distributed transactions, make sure that you relate the tests to a single Business Process Monitor instance. Business Process Monitor searches for the end transaction name in all instances, and may close the wrong distributed transaction if it is included in more than one instance.

- When measuring a distributed transaction over two Business Process Monitor profiles, make sure that the timeout value you specify is large enough so that the profile or Business Transaction Flow (depending on the version) that contains the **StartDistributedTransaction** step and all the profiles that run before the profile that contains the **EndDistributedTransaction** step, will finish running in a time that is less than the value of the specified timeout.

Running GUI Tests from HP Business Process Monitor

Relevant for: GUI tests only

Consider the following guidelines when running tests from HP Business Process Monitor:

- Before you try to run a test in Business Process Monitor, check which versions of UFT are supported by your version of Business Process Monitor. For details, see the Business Process Monitor documentation.

- To run a test in Business Process Monitor, UFT must be installed and closed on the Business Process Monitor computer

- Business Process Monitor can run only one test at a time. Make sure that the previous UFT run session is finished before starting to run another test.

- Transaction breakdown is not supported for tests created with UFT.

- Tests must be zipped before uploading them to Business Service Management Admin.

  If you make changes to your local copy of a test after uploading it to Business Service Management, upload the zipped test again to enable Business Process Monitor to run the test with your changes.

- UFT cannot run tests on a computer that is logged off, locked, or running UFT as a non-interactive service.

- You should start Business Process Monitor by running the magentproc.exe program when running tests from Business Process Monitor.

- You cannot use the **ResultDir** environment variable when running a test in Business Process Monitor.

For details on working with Business Service Management, see the relevant documentation—specifically the sections describing UFT scripts in the Business Process Monitor Administration guide and the End User Management guide.

> **Tip:** You can simulate how the test will run from Business Process Monitor by using Silent Test Runner. For details, see "Silent Test Runner" on page 2239.

Measuring Transactions

Relevant for: GUI tests only

You can measure how long it takes to run a section of your test by defining **transactions**. A transaction represents the process in your application that you are interested in measuring. Your test must include transactions to be used by LoadRunner, Performance Center, or the Business Process Monitor. These products use only the data that is included within a transaction, and ignore any data in a test outside of a transaction.

You define transactions within your test by enclosing the appropriate sections of the test with **start** and **end** transaction statements. For example, you can define a transaction that measures how long it takes to reserve a seat on a flight and for the confirmation to be displayed on the client's terminal.

During the run session, the **StartTransaction** step signals the beginning of the time measurement. The time measurement continues until the **EndTransaction** step is reached. The test results for the EndTransaction step include the transaction's name, end status, total duration, and wasted time.

During a run session, UFT runs background processes that add to the time it takes to run a test. Wasted time is the time within the total duration that was added as a result of UFT running the transaction. If the application ran the transaction without UFT, the total duration would equal the total duration minus the wasted time.

> **Note:** If you start a transaction while there is already open transaction with the same name, the previous transaction is ended with **Fail** status and then the new transaction is started.

There is no limit to the number of transactions that can be added to a test.

> **Tip:**

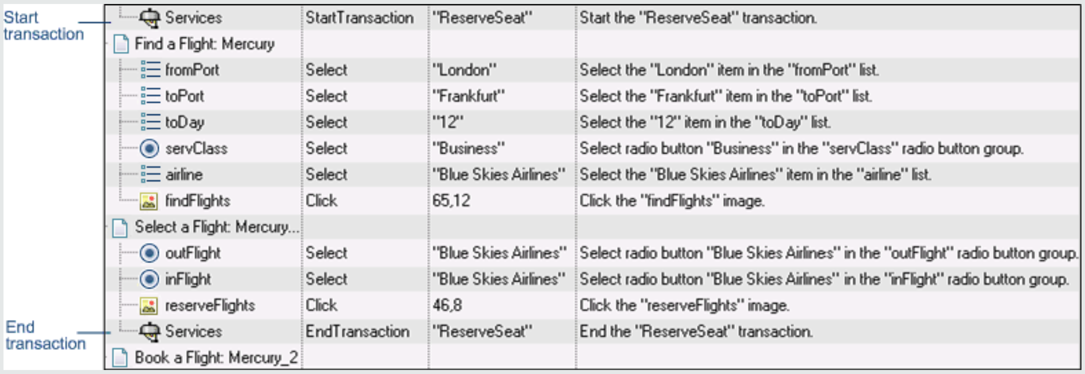
You can:

- Insert a transaction within a transaction.
- Insert a variety of transaction-related statements using the Step Generator or Editor. For details, see the **Services** section of the *HP UFT Object Model Reference for GUI Testing*.
- Enter Start Transaction and End Transaction steps using UFT. For details, see "Start Transaction Dialog Box" on page 2243 and "End Transaction Dialog Box" on page 2242.

For details on the statements you can use in transactions, see the *HP UFT Object Model Reference for GUI Testing*.

Example of a test with a transaction

Part of a sample test with a transaction is shown below, as it is displayed in the Keyword View:



Services	StartTransaction	"ReserveSeat"	Start the "ReserveSeat" transaction.
Find a Flight: Mercury			
fromPort	Select	"London"	Select the "London" item in the "fromPort" list.
toPort	Select	"Frankfurt"	Select the "Frankfurt" item in the "toPort" list.
toDay	Select	"12"	Select the "12" item in the "toDay" list.
servClass	Select	"Business"	Select radio button "Business" in the "servClass" radio button group.
airline	Select	"Blue Skies Airlines"	Select the "Blue Skies Airlines" item in the "airline" list.
findFlights	Click	65,12	Click the "findFlights" image.
Select a Flight: Mercury...			
outFlight	Select	"Blue Skies Airlines"	Select radio button "Blue Skies Airlines" in the "outFlight" radio button group.
inFlight	Select	"Blue Skies Airlines"	Select radio button "Blue Skies Airlines" in the "inFlight" radio button group.
reserveFlights	Click	46,8	Click the "reserveFlights" image.
Services	EndTransaction	"ReserveSeat"	End the "ReserveSeat" transaction.
Book a Flight: Mercury_2			

The same part of the test is displayed in the Editor as follows:

```
Services.StartTransaction "ReserveSeat"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").   WebList("fromPort").Select "London"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").   WebList("toPort").Select "Frankfurt"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").   WebList("toDay").Select "12"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").   WebRadioGroup("servClass").Select "Business"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").   WebList("airline").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").   Image("findFlights").Click 65,12
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").   WebRadioGroup("outFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").   WebRadioGroup("inFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").   Image
```

```
("reserveFlights").Click 46,8
Services.EndTransaction "ReserveSeat"
```

Silent Test Runner

Relevant for: GUI tests only

Silent Test Runner enables you to simulate the way a test runs from LoadRunner, Performance Center, and Business Service Management. When you run a test using Silent Test Runner, it runs without opening the UFT user interface, and the test runs at the same speed as when it is run from LoadRunner, Performance Center, or Business Service Management At the end of the test run, you can view information about the test run and transaction times. For details, see "Test Run Information for Silent Runs" below.

You can also use Silent Test Runner to verify that your test is compatible with LoadRunner, Performance Center, and Business Service Management. A test will fail when run using Silent Test Runner if it uses a feature that is not supported by these products. For details on features that are not supported, see "Designing Tests for HP Performance Testing Products" on page 2234, "Running GUI Tests from HP Performance Testing Products " on page 2234, "Designing GUI Tests for HP Business Process Monitor" on page 2235, and "Running GUI Tests from HP Business Process Monitor" on page 2236.

Test Run Information for Silent Runs

Silent Test Runner provides test run information in log files. Each test generates a test run log, and any test with transactions generates an additional transaction summary.

Viewing the Test Run Log

The test run log is saved as output.txt in the <**Unified Functional Testing**>\Tests\<test name> folder. A log file is saved for each test run with Silent Test Runner and is overwritten when you rerun the test. To open the log file, click **Test Run Log**.

The log file displays information about the test run. For example, information is shown about each iteration, action call, step transaction, failed step, and so forth. Each line displays a message or error ID. For details on message and error codes in the log file, see your Performance Center or Business Service Management documentation.

Viewing the Transaction Summary

The transaction summary is saved as transactions.txt in the <Unified Functional Testing>\Tests\<test name> folder. A transaction summary is saved for each test that includes transactions and is overwritten when you rerun the test. To open the log file, click **Transaction Summary**. The transaction summary displays a line for each transaction in the test. For each transaction, the status is displayed together with the total duration time and any wasted time (in seconds). The transaction measurements in Silent Test Runner are exactly the same as if the test was run from LoadRunner, Performance Center, or Business Service Management.

Note:

- A transaction summary is available only for a test that contains transactions ending with an **EndTransaction** statement. If a transaction started but did not end because of test failure, it is not included in the transaction summary.
- Distributed transactions (transactions that start in one test and end in another) are not reported in the transaction summary but are included in the test run log.
- Any transaction information included in the transaction summary is also included in the test run log.

Tasks

How to Insert and Run GUI Tests in Performance Center and LoadRunner

Relevant for: GUI tests only

The following are various tasks that you can perform when working with HP Performance Center and LoadRunner.

To insert a test in a LoadRunner scenario

In the Controller Open Test dialog box, browse to the test folder and select **QuickTest Tests** or **GUI Scripts** (for tests created in the SAP environment) in the **Files of type** box. This enables you to view the tests in the folder.

To use a UFT test in Performance Center

Create a zipped version of the test, and upload it to the Performance Center User Site Vuser Scripts Page.

To run multiple GUI Vusers on the same application

Open a terminal server session for each GUI Vuser. For details, refer to the HP performance testing documentation.

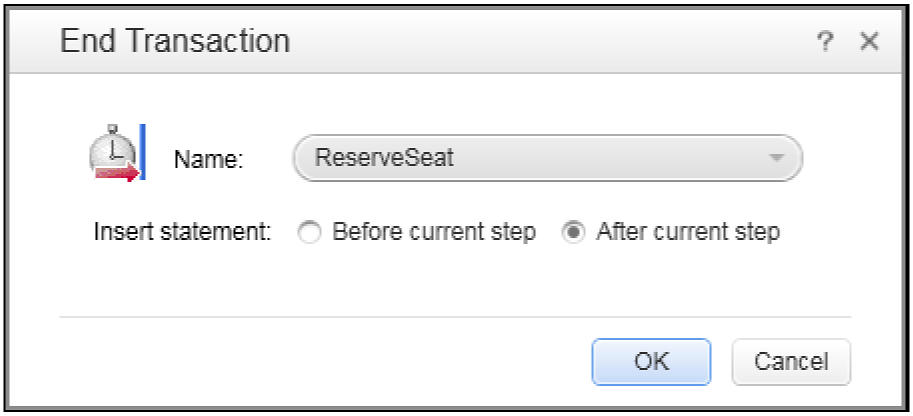
For details on working with LoadRunner or Performance Center, see your HP performance testing documentation.

Reference

End Transaction Dialog Box

Relevant for: GUI tests only

This dialog box enables you to insert a step that signals the end of the time measurement for a transaction.



End Transaction

Name: ReserveSeat

Insert statement: ○ Before current step ● After current step

To access	1. Open the existing test into which you want to insert a load transaction step. 2. Select the **Design > End Transaction**.
Important information	- There may be cases in which you want to instruct UFT to perform all the steps in a transaction, even though an error occurs during the run session. To do this: In the Run pane of the Test Settings dialog box (**File > Settings > Run** node), select **proceed to next step** from the **When error occurs during run session** list. - You can also create recovery scenarios or other error handling steps to address these cases. For details, see "Recovery Scenarios for GUI Testing" on page 1111.
See also	- "Measuring Transactions" on page 2237 - "Start Transaction Dialog Box" on the next page

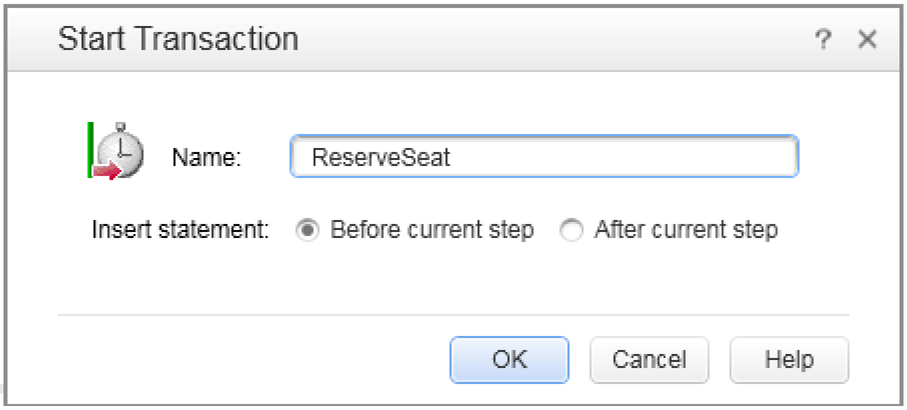
User interface elements are described below:

UI Elements	Description
Name	The name of the transaction you want to end. The list contains the name of all transactions that start prior to the selected step in the current action.
Insert Statement	Indicates where the **EndTransaction** step will be inserted in relation to the selected step. Select **Before current step** or **After current step**.

Start Transaction Dialog Box

Relevant for: GUI tests only

This dialog box enables you to insert a step that signals the beginning of the time measurement for a transaction.



Name: ReserveSeat

Insert statement: ● Before current step ○ After current step

To access	1. Open the existing test into which you want to insert a load transaction step. 2. Select the **Design > Start Transaction** menu command.
See also	- "Measuring Transactions" on page 2237 - "End Transaction Dialog Box" on the previous page

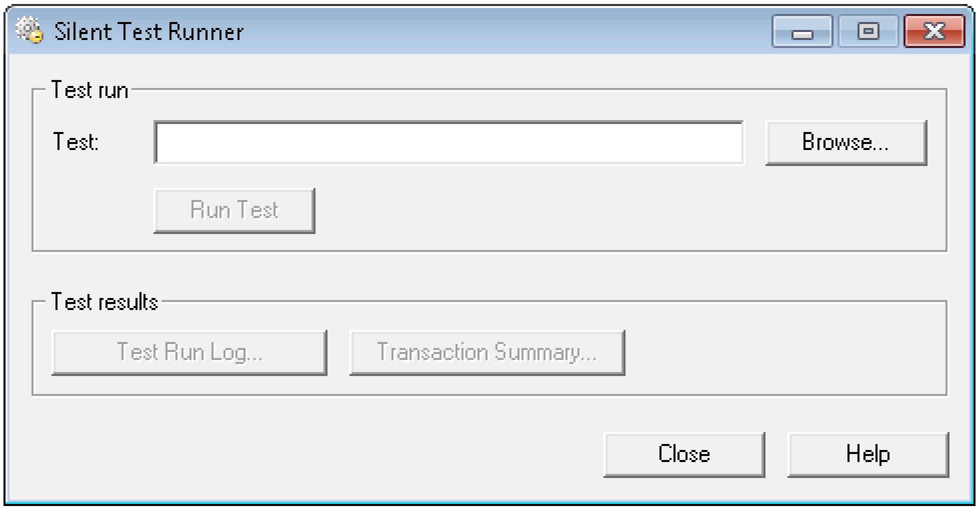
User interface elements are described below:

UI Elements	Description
Name	The name of the transaction you want to measure. **Note:** You cannot include spaces in a transaction name.
Insert Statement	Indicates where the **StartTransaction** step will be inserted in relation to the selected step. You can select **Before current step** or **After current step**.

Silent Test Runner Dialog Box

Relevant for: GUI tests only

This dialog box enables you to simulate the way a UFT test runs from LoadRunner and Business Service Management and to verify that your UFT test is compatible with LoadRunner and Business Service Management.



To access	Select the **Start > Programs > HP Software > HP Unified Functional Testing > Tools > Silent Test Runner** menu command. **Note:** For details on accessing UFT and UFT tools and files in Windows 8, see "Accessing UFT in Windows 8 Operating Systems" on page 75.
Important information	• You can invoke only one instance of Silent Test Runner and you can specify only one test to run. • You cannot use the **ResultDir** environment variable when running a test from Silent Test Runner.

See also	- "Silent Test Runner" on page 2239 - "Test Run Information for Silent Runs" on page 2239

User interface elements are described below:

UI Elements	Description
Test	The full file system path of the test you want to run. **Note:** To specify a network path, you must map the network drive.
Run Test	Runs the test specified in the **Test** box. When you click this button, the test runs without opening the UFT user interface. The text **Running test...** is displayed next to the **Run Test** button while the test is running. When the test run finishes, the text **Running test...** is replaced with the text **Test run completed**. If Silent Test Runner was unable to run your test, the text **Test could not be run** is displayed. **Note:** After you start a test run, you cannot stop the test run from Silent Test Runner. Even if you close Silent Test Runner, the test continues to run. To end the run, end the mdrv.exe process manually.
Test Run Log	Displays the most recent run log for the selected test. Each time you run a test with Silent Test Runner, the previous log file is overwritten with the current run results. (Enabled only when the selected test has run with the Silent Test Runner at least once.) For details, see "Viewing the Test Run Log" on page 2239.
Transaction Summary	Displays the summary of the transactions in the test. (Enabled only when the selected test contains at least one transaction and the test has run with the Silent Test Runner at least once.) For details, see "Viewing the Transaction Summary" on page 2240.

Appendix H: Frequently Asked Questions for GUI Testing

Relevant for: GUI tests and components

This chapter answers some of the questions that are asked most frequently by advanced users of UFT. The questions and answers are divided into the following sections:

This chapter includes:

Creating Tests or Components

Relevant for: GUI tests and components

This section includes answers to the following questions:

- "How can I record on objects or environments not supported by UFT?" below
- "How can I launch an application from a test or component?" on the next page
- "How does UFT capture user processes in Web pages?" on the next page
- "Can I insert calls to WinRunner tests and functions from UFT?" on the next page

How can I record on objects or environments not supported by UFT?

You can do this in a number of ways:

- Install and load any of the add-ins that are available for UFT. UFT supports many developmental environments including Java, Oracle, .NET, SAP Solutions, Siebel, PeopleSoft, and terminal emulators.
- You can map objects of an unidentified or custom class to standard Windows classes. For details on object mapping, see "Test Object Mapping for Unidentified or Custom Classes" on page 1321.
- You can use the Insight recording mode. For details, see "Insight Recording" on page 849.
- UFT provides add-in extensibility that you can use to extend UFT built-in support for various objects. This enables you to direct UFT to recognize an object as belonging to a specific test object class, and to specify the behavior of the test object. You can also extend the list of available test object classes that UFT recognizes. This enables you to create tests or components that fully support the specific behavior of your custom objects.
- **For tests and scripted components:**
  - You can define virtual objects for objects that behave like test objects, and then record in the normal recording mode. For details on defining virtual objects, see "Virtual Objects" on page 1384.
  - You can record your clicks and keyboard input based on coordinates in the low-level or analog recording modes. For details on low-level and analog recording, see "Recording Modes" on page 845.

How can I launch an application from a test or component?

To launch an application from within a test or scripted component:

Add a **SystemUtil** step, such as:

```
SystemUtil.Run "D:\My Music\Breathe.mp3","","D:\My Music\Details","open"
```

For Windows-based applications, you should also ensure that in the Windows Applications tab of the Record and Run Settings dialog box, you configure UFT to record and run on applications opened by UFT.

To launch an application from within a keyword component:

Select **Operation** from the **Item** column, select **OpenApp** from the **Operation** column, and then enter the full path in the **Value** column, for example:

```
%ProgramFiles%\HP\Unified Functional Testing\samples\flight\app\flight4a.exe
```

How does UFT capture user processes in Web pages?

UFT hooks the Microsoft Internet Explorer browser. As the user navigates the Web-based application, UFT records the user operations. (For details on modifying which user operations are recorded, see the section on configuring Web event recording in the *HP Unified Functional Testing Add-ins Guide*.) UFT can then run the test or component by running the steps as they originally occurred.

Can I insert calls to WinRunner tests and functions from UFT?

The **Insert > Call to WinRunner > Test** and **Insert > Call to WinRunner > Function** commands are no longer available. You can continue to run existing UFT business process tests that contain calls to WinRunner tests or components and functions, and you can view run results in the Run Results Viewer.

> **Note:** Hewlett-Packard (HP) has discontinued HP WinRunner (WR) 7.5, 7.6, 8.0, 8.2, 9.2 (all editions) and has announced an End-of-Support timeline for these products. For details, see: http://support.openview.hp.com/encore/wr.jsp?jumpid=reg_R1002_USEN

Programming in the Editor and Working with Function Libraries

Relevant for: GUI tests and components

This section includes answers to the following questions:

- "Can I store functions and subroutines in a function library?" on the next page
- "How can I enter information during a run session?" on the next page

- "For tests: I have a Microsoft Access database that contains data I would like to use in my test. How do I do this?" below
- "How do I customize the Run Results?" on the next page

Can I store functions and subroutines in a function library?

You can create one or more VBScript function libraries containing your functions. You can then call the functions from any test, or use them in any component, by associating them with the test or with the component's application area. You can use the UFT function library editor to create and debug your function libraries.

You can also register your functions as methods for UFT test objects. Your registered methods can override the functionality of an existing test object method for the duration of a run session, or you can register a new method for a test object class.

For more details, see "User-Defined Functions and Function Libraries" on page 1029, and, for components, "Application Areas" on page 2095.

For tests: You can define functions within an individual action, making them available for use inside the action. If you save the action as a reusable action, it functions are available in the actions that call it as well. However, you can help improve UFT performance by storing your functions in function libraries instead of in reusable actions.

How can I enter information during a run session?

The VBScript **InputBox** function enables you to display a dialog box that prompts the user for input, and then continues running the test or component. You can use the value that was entered by the user later in the run session. For details on the **InputBox** function, see the *VBScript Reference*.

For components: You can insert the VBScript **InputBox** function into an associated function library to create a user-defined VBScript **InputBox** function.

For tests: The following example shows the **InputBox** function used to prompt the user for a password:

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set "administrator"
Passwd = InputBox ("Enter password", "User Input")
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("password").Set Passwd
```

For tests: I have a Microsoft Access database that contains data I would like to use in my test. How do I do this?

The Editor enables you to access databases using ADO and ODBC. Below is a sample test that searches for books written by an author in the "Authors" table of the database.

```
Dim MyDB
Dim MyEng
Set MyEng = CreateObject("DAO.DBEngine.35")
Dim Td
Dim rs
' Specify the database to use.
Set MyDB = MyEng.OpenDatabase("BIBLIO.MDB")
' Read and use the name of the first 10 authors.
Set Td = MyDB.TableDefs("Authors")
Set rs = Td.OpenRecordset
rs.MoveFirst
For i = 1 To 10
   Browser("Book Club").Page("Search Books").WebEdit("Author Name").Set rs("Author")
   Browser("Book Club").Page("Search Books").WebButton("Search").Click
Next
```

How do I customize the Run Results?

You can add information to the run results by using the **ReportEvent** method. This adds a step to the run results containing a free-text message and a step status that can potentially affect the status of the run session. The step can also include an image, such as a logo, if you specify an image file path.

For example:

```
Reporter.ReportEvent 1, "Custom Step", "The user-defined step failed"
```

For more details, see the *HP UFT Object Model Reference for GUI Testing*.

The results of each UFT run session are saved in a single .xml file (called results.xml). You can modify this file, as needed. You can use the *HP Run Results Schema Reference* (**Help > HP UFT GUI Testing Advanced References Help > HP Run Results Schema Reference**) to help you customize your run results.

Working with Dynamic Content

Relevant for: GUI tests and components

This section includes answers to the following questions:

- "How can I create and run tests or components on objects that change dynamically from viewing to viewing?" below
- "How can I check that an object or child object exists (or does not exist)?" on the next page
- "How does UFT record on dynamically generated URLs and Web pages?" on the next page
- "How does UFT handle tabs in browsers?" on page 2252

How can I create and run tests or components on objects that change dynamically from viewing to viewing?

Sometimes the content of objects in an application changes due to dynamic content. You can create dynamic descriptions of these objects so that UFT will recognize them when it runs the test or component using regular expressions, the **Description** object, repository parameters, or **SetTOProperty** steps.

How can I check that an object or child object exists (or does not exist)?

Some objects are created in an application only after you perform an operation. For example, a link in one window sometimes creates another window. The newly created window may be an independent object, or a child of the original window.

Before you perform operations on an object created during a run session, you may want to verify that the object already exists.

If you have a test object that corresponds to the object in the application, you can use the **Exist** property to check whether the object exists in the application. The **Exist** property looks for an object in the application that matches the test object's description. For example:

```
If Window("Main").ActiveX("Slider").Exist Then
. . .
```

Alternatively, you can use the **ChildObjects** method to retrieve all child objects (or the subset of child objects that match a certain description) on the Desktop or within any other parent object.

Example

```
Set oDesc = Description.Create
oDesc("Class Name").Value = "Window"
Set coll = Desktop.ChildObjects(oDesc)
For i = 0 to coll.count -1
      msgbox coll(i).GetROProperty("text")
Next
```

> **Note:**
>
> - After you use the **ChildObjects** method to retrieve an object, UFT accesses the object directly in the application and does not save a description for the object. Therefore, you must use the object immediately after retrieving it, before anything in the application changes.
>
> - The **Exist** property searches for objects based on their description, and is therefore not relevant for objects retrieved by the **ChildObjects** method. (The **Exist** property always returns true when called for such objects).

For more details on the **Exist** property and **ChildObjects** method, see the *HP UFT Object Model Reference for GUI Testing*.

How does UFT record on dynamically generated URLs and Web pages?

UFT actually clicks links as they are displayed on the page. Therefore, UFT records how to find a particular object, such as a link on the page, rather than the object itself. For example, if the link to a dynamically generated URL is an image, then UFT records the "IMG" HTML tag, and the name of the image. This enables UFT to find this image in the future and click on it.

How does UFT handle tabs in browsers?

UFT provides several methods that you can use with the **Browser** test object to manage tabs in your Web browser.

OpenNewTab opens a new tab in the current Web browser.

IsSiblingTab indicates whether a specified tab is a sibling of the current tab object in the same browser window.

Close closes the current tab if more than one tab exists, and closes the browser window if the browser contains only one tab.

CloseAllTabs closes all tabs in a browser and closes the browser window.

For more details on these **Browser**-related methods, see the **Web** section of the *HP UFT Object Model Reference for GUI Testing*.

Advanced Web Issues

Relevant for: GUI tests and components

This section includes answers to the following questions:

- "How does UFT handle cookies? " below
- "Where can I find a Web page's cookie?" on the next page
- "How does UFT handle session IDs? " on the next page
- "How does UFT handle server redirections? " on the next page
- "How does UFT handle meta tags? " on the next page
- "Does UFT work with .asp and .jsp?" on the next page
- "How does UFT support AJAX?" on the next page
- "Does UFT work with COM?" on the next page
- "Does UFT work with XML?" on the next page
- "How can I access HTML tags directly?" on page 2254
- "Where can I find information on the Internet Explorer Document Object Model?" on page 2254
- "How can I send keyboard key commands (such as shortcut commands) to objects that do not support the Type method?" on page 2254

How does UFT handle cookies?

Server-side connections, such as CGI scripts, can use cookies both to store and retrieve information on the client side of the connection.

UFT stores cookies in the memory for each user, and the browser handles them as it normally would.

Where can I find a Web page's cookie?

The cookie used by the Internet Explorer browser can be accessed through the browser's Document Object Model (DOM) using the **.Object** property (for components, you do this in a user-defined function). In the following example the cookie collection is returned the from the browser:

```
Browser("Flight reservations").Page("Flight reservations").Object.Cookie
```

How does UFT handle session IDs?

The server, not the browser, handles session IDs, usually by a cookie or by embedding the session ID in all links. This does not affect UFT.

How does UFT handle server redirections?

When the server redirects the client, the client generally does not notice the redirection, and misdirections generally do not occur. In most cases, the client is redirected to another script on the server. This additional script produces the HTML code for the subsequent page to be viewed. This has no effect on UFT or the browser.

How does UFT handle meta tags?

Meta tags do not affect how the page is displayed. Generally, they contain information only about who created the page, how often it is updated, what the page is about, and which keywords represent the page's content. Therefore, UFT has no problem handling meta tags.

Does UFT work with .asp and .jsp?

Dynamically created Web pages utilizing Active Server Page technology have an **.asp** extension. Dynamically created Web pages utilizing Java Server Page technology have a **.jsp** extension. These technologies are completely server-side and have no bearing on UFT.

How does UFT support AJAX?

You can use UFT Web Add-in Extensibility to add your own support for custom Web controls. The Web Add-in Extensibility SDK installs a sample toolkit support set that provides partial support for some ASP .NET AJAX controls. You can use this sample to learn how to create your own support for your AJAX controls. For more details, see the *HP UFT Web Add-in Extensibility Developer Guide*.

Does UFT work with COM?

UFT complies with the COM standard.

UFT supports COM objects embedded in Web pages (which are currently accessible only using Microsoft Internet Explorer), and you can drive COM objects in VBScript.

Does UFT work with XML?

XML is eXtensible Markup Language, a pared-down version of SGML for Web documents, that enables Web designers to create their own customized tags. UFT supports XML and recognizes

XML tags as objects.

For tests and scripted components: You can also create XML checkpoints to check the content of XML documents in Web pages, frames or files. UFT also supports XML output and schema validation.

For more details, see "XML Checkpoints Overview" on page 1416, and the **XMLUtil** object in the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

How can I access HTML tags directly?

UFT provides direct access to the Internet Explorer's Document Object Model (DOM) through which you can access the HTML tags directly. Access to the DOM is performed using the .Object notation.

The function below demonstrates how to iterate over all the tags in an Internet Explorer page. The function then outputs the inner-text of the tags (the text contained between the tags) to the Run Results using the **Reporter** object.

```
' Use the on error option because not all the elements have inner-text.
On Error Resume Next
Set Doc = Browser("CNN Interactive").Page("CNN Interactive").Object
' Loop through all the objects in the page.
For Each Element In Doc.all
   TagName = Element.TagName ' Get the tag name.
   InnerText = Element.innerText ' Get the inner text.
   ' Write the information to the run results.
   Reporter.ReportEvent 0, TagName, InnerText
Next
```

Where can I find information on the Internet Explorer Document Object Model?

For details on the Internet Explorer DOM, browse to the following Web sites:

Document object:
http://msdn.microsoft.com/en-us/library/ms531073.aspx

Other DHTML objects:
http://msdn.microsoft.com/en-us/library/ms533054.aspx

General DHTML reference:
http://msdn.microsoft.com/en-us/library/ms533050.aspx

How can I send keyboard key commands (such as shortcut commands) to objects that do not support the Type method?

For objects that do not support the **Type** method, use the Windows Scripting **SendKeys** method. For more details, see the Microsoft VBScript Language Reference (choose **Help > HP Unified Functional Testing Help > VBScript Reference > Windows Script Host**).

Standard Windows Environment

Relevant for: GUI tests and components

This section includes the answers to the following questions:

- "How can I record on nonstandard menus?" below
- "For tests: How can I terminate an application that is not responding?" below
- "For tests: Can I copy and paste to and from the Clipboard during a run session?" below

How can I record on nonstandard menus?

You can modify how UFT behaves when it records menus. The options that control this behavior are located in the Windows Applications > Advanced Options pane. (**Tools > Options > GUI Testing** tab **> Windows Applications** node **> Advanced** node).

For more details, see the *HP Unified Functional Testing Add-ins Guide*.

For tests: How can I terminate an application that is not responding?

You can terminate any standard application while running a test in UFT by adding one of the following steps to the test:

- SystemUtil.CloseProcessByName "<app.exe>"
- SystemUtil.CloseProcessByWndTitle "<Some Title>"

For tests: Can I copy and paste to and from the Clipboard during a run session?

You can use the Clipboard object to copy, cut, and paste text during a UFT run session.

The Clipboard object supports the same methods as the Clipboard object available in Visual Basic, such as:

- Clear
- GetData
- GetText
- SetData
- SetText

For details on these methods, see http://msdn.microsoft.com/en-us/library/ms172962.aspx.

Below is an example of Clipboard object usage:

```
Set MyClipboard = CreateObject("Mercury.Clipboard")
```

```
MyClipboard.Clear
MyClipboard.SetText "TEST"
MsgBox MyClipboard.GetText
```

Test and Component Maintenance

Relevant for: GUI tests and components

This section includes answers to the following questions:

- "How do I maintain my test or component when my application changes?" below
- "For tests and scripted components: Can I increase or decrease Active Screen information after I finish recording a test?" on the next page
- "For tests: How can I remove run results files from old tests?" on the next page

How do I maintain my test or component when my application changes?

The way to maintain a test or component when your application changes depends on how much your application changes. This is one of the main reasons you should create a small group of tests or components, rather than one large test or component for your entire application.

For tests: You can also use UFT actions to design more modular and efficient tests. Divide your test into several actions, based on functionality. When your application changes, you can modify a specific action, without changing the rest of the test. Whenever possible, insert calls to reusable actions rather than creating identical pieces of script in several tests. This way, changes to your original reusable action are automatically applied to all tests calling that action. For more details, see "Actions in GUI Testing" on page 871.

If you have many tests, components, and actions that contain the same test objects, it is recommended to work with shared object repositories so that you can update object information in a centralized location.

You can use the **Update Run Mode** option to update changed information for checkpoints or the Active Screen, or to change the set of identification properties used to identify the objects in your application. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

If there is a discrepancy between the identification property values saved in the object repository and the object property values in the application, you can use the **Maintenance Run Mode** to help correct this. When you run a test or component in Maintenance Run Mode, UFT runs your test or component, and then guides you through the process of updating your steps and object repository each time it encounters a step it cannot perform due to an object repository discrepancy. For more details, see "Maintenance Run Mode" on page 1080.

For tests and scripted components: Can I increase or decrease Active Screen information after I finish recording a test?

If you find that the information saved in the Active Screen after recording is not sufficient, or if you no longer need Active Screen information, and you want to decrease the size of your test or component, there are several methods of changing the amount of Active Screen information saved.

- To decrease the disk space used by your test or component, you can delete Active Screen information by selecting **Save As**, and clearing the **Save Active Screen files** check box. For details, see "Save <Resource>/Save <Document> As Dialog Box" on page 164.

- If you chose not to save all information in the Active Screen when testing a Windows application, you can use one of several methods to increase the information stored in the Active Screen.

  Confirm that the Active Screen capture preference in the **Active Screen** pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Active Screen** node) is set to capture the amount of information you need and then:

  - Perform an **Update Run Mode** operation to save the required amount of information in the Active Screen for all existing steps. For details on the **Update Run Mode** options, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

  - Re-record the steps containing the objects you want to add to the Active Screen.

    To re-record the step, select the step after which you want to record your step, position your application to match the selected location in your test or component, and then begin recording. Alternatively, place a breakpoint in your test at the step before which you want to add a step and run your test or component to the breakpoint. This brings your application to the point from which to record the step. For details on setting breakpoints, see "Breakpoints " on page 681.

For more details on changing the amount of information saved in the Active Screen for Windows applications, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

For tests: How can I remove run results files from old tests?

You can use the Run Results Deletion Tool to view a list of all of the run results in a specific location in your file system or in your ALM project. You can then delete any run results that you no longer require.

The Run Results Deletion Tool enables you to sort the run results by name, date, size, and so forth, so that you can more easily identify the results you want to delete.

To open this utility, choose **Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Run Results Deletion Tool**.

Testing Localized Applications

Relevant for: GUI tests only

I am testing localized versions of a single application, each with localized user interface strings. How do I create efficient tests in UFT?

You can parameterize these user interface strings using parameters from the global Environment variable list. This is a list of variables and corresponding values that can be accessed from any test. For details, see "Parameterizing Object Values" on page 1526.

I am testing localized versions of a single application. How can I efficiently input different data in my tests, depending on the language of the application?

If you are running a single iteration of your test, or if you want values to remain constant for all iterations of an action or test, use environment variables, and then change the active environment variable file for each test run.

If you are running multiple iterations of your test or action, and you want the input data to change in each iteration, you can create an external data table for each localized version of your application. When you change the localized version of the application you are testing, you simply switch the data table file for your test in the Resources pane of the Test Settings dialog box. For details on working with data tables, see "Data Pane" on page 221. For details on selecting the data table file for your test, see "Resources Pane (Test/Business Component Settings Dialog Box) " on page 603.

Improving GUI Testing Performance

Relevant for: GUI tests and components

This section includes the answers to the following questions:

- "How can I improve the working speed of UFT when working with GUI testing?" below
- "How can I decrease the disk space used by UFT for GUI tests and components? " on page 2260
- "For tests: Is there a recommended length for tests?" on page 2261

How can I improve the working speed of UFT when working with GUI testing?

You can improve the working speed of UFT by doing any of the following:

- In the Add-in Manager, load only the add-ins you need for a specific UFT session when UFT starts. This will improve performance while learning objects and during run sessions. For details on loading add-ins, see the *HP Unified Functional Testing Add-ins Guide*.
- Run your tests or components in "fast mode." From the **Test Runs** pane in the Options dialog

box (**Tools > Options > GUI Testing** tab **> Test Runs** node), select the **Fast** option. This instructs UFT to run your test or component without displaying the execution arrow for each step, enabling the test or component to run faster. For details on the Test Runs pane of the Options dialog box, see "Test Runs Pane (Options Dialog Box > GUI Testing Tab)" on page 539.

- Decide if and when you want to capture and save images and/or movies of the application for the run results. You can reduce disk space and improve test run time by saving screen captures and movie segments only in certain situations, such as when errors occur, or by not saving them at all. To do this, use the **Save still image captures to results** and **Save movie to results** options in the **Screen Capture** pane in the Options dialog box (**Tools > Options > GUI Testing** tab **> Screen Capture** node). For details, see "Screen Capture Pane (Options Dialog Box > GUI Testing Tab)" on page 557.

- If you are using Insight test objects, adjust the number and size of snapshots saved with the test objects. For details, see "Insight Pane (Options Dialog Box > GUI Testing Tab)" on page 561.

  In the object repository, you can delete all of the snapshots stored with the Insight test objects after you finalize the test object images and verify that they enable correct object identification in all relevant scenarios. (In the Object Repository window or the Object Repository Manager, **Tools > Delete Insight Snapshots**.)

- Save the run results report to a temporary folder to overwrite the results from the previous run session every time you run a test or component. For more details, see "Run Dialog Box" on page 651.

- **For components:** Try to use the same application area for all components in a business process test, as this improves performance.

- **For tests:** Minimize the number of actions in a test. Ideally, a test should not contain more than a few dozen actions.

- **For tests:** Store your functions in function libraries instead of as reusable actions.

- **For tests:** Use the Results Deletion Tool to remove unwanted or obsolete run results from your system, according to specific criteria that you define. This enables you to free up valuable disk space. For details, see the section on the Run Results Deletion Tool (described in the *HP Run Results Viewer User Guide*).

- **For tests and scripted components:** If you are not using the Active Screen while editing your test, hide the Active Screen while editing your test to improve editing response time by right-clicking the Active Screen pane and selecting **Hide**. For details, see "UFT at a Glance" on page 77.

- **For tests and scripted components:** Decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test or component using the many Active Screen options, but more captured information also leads to slower recording and editing times. You can choose from the following Active Screen options to improve performance:

  - If you are testing Windows applications, you can choose to save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the **Active Screen** pane of the Options dialog box. For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

  - If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the **Active Screen** pane of the Options dialog box, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box.

    Select the **Disable Active Screen Capture** option. This will improve recording time. For details on the Active Screen pane of the Options dialog box (**Tools > Options > GUI Testing** tab **> Active Screen** node), see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

  - When you save a new test or component, or when you save a test or component with a new name using **Save As**, you can choose not to save the captured Active Screen files with the test or component by clearing the **Save Active Screen files** option in the Save or Save As dialog box. This is especially useful when you have finished designing your test or component and you plan to use your it only for run sessions. Tests and components without Active Screen files open more quickly and use significantly less disk space.

    For details on the Active Screen pane of the Options dialog box, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

  > **Tip:** (For tests and scripted components) If you need to recover Active Screen files after you save a test or component without Active Screen files, re-record the necessary steps or use the **Update Run Mode** option to recapture screens for all steps. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

How can I decrease the disk space used by UFT for GUI tests and components?

You can decrease the disk space used by UFT by doing any of the following:

- Decide if and when you want to capture and save images and/or movies of the application for the run results. You can reduce disk space and improve test run time by saving screen captures and movie segments only in certain situations, such as when errors occur, or by not saving them at all. To do this, use the **Save still image captures to results** and **Save movie to results** options in the **Screen Capture** pane in the Options dialog box (**Tools > Options > GUI Testing** tab **> Screen Capture** node). For details, see "Screen Capture Pane (Options Dialog Box > GUI Testing Tab)" on page 557.

- If you are using Insight test objects, adjust the number and size of snapshots saved with the test objects. For details, see "Insight Pane (Options Dialog Box > GUI Testing Tab)" on page 561. These settings can also affect UFT performance when recording and running Insight steps.

  In the object repository, you can delete all of the snapshots stored with the Insight test objects after you finalize the test object images and verify that they enable correct object identification in all relevant scenarios.

- **For tests and scripted components:** Decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options, but more captured information also leads to slower recording and editing times. You can choose from the following Active Screen options to improve performance:

  - If you are testing Windows applications, you can choose to Save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the **Active Screen** pane of the Options dialog box. For details, see "Active Screen Pane (Options Dialog Box > GUI Testing Tab)" on page 547.

  - If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen pane, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box. Select the **Disable Active Screen Capture** option. This will improve recording time. For details on the Active Screen pane of the Options dialog box, see "Screen Capture Pane (Options Dialog Box > GUI Testing Tab)" on page 557.

  - When you save a new test, or when you save a test with a new name using Save As, you can choose not to save the captured Active Screen files with the test by clearing the **Save Active Screen files** option in the Save or Save As dialog box. This is especially useful when you have finished designing your test and you plan to use your test only for test runs. Tests without Active Screen files use significantly less disk space.

  Tip: (For tests and scripted components) If you need to recover Active Screen files after you save a test without Active Screen files, re-record the necessary steps or use the **Update Run Mode** option to recapture screens for all steps in your test. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1107.

For tests: Is there a recommended length for tests?

Although there is no formal limit regarding test length, it is recommended that you divide your tests into actions and that you use reusable actions in tests, whenever possible. An action should contain no more than a few hundreds steps and, ideally, no more than a few dozen. For details, see "Actions in GUI Testing" on page 871.

Note: See also "Improving the working speed (performance) of UFT" on page 2264

Appendix I: UFT Troubleshooting and Limitations

Relevant for: GUI tests and components and API testing

This chapter includes:

Troubleshooting and Limitations - UFT Program Management

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for UFT tools and program management, and covers the following topics:

- "Tools in the HP UFT program group" below
- "Characters do not display correctly" on the next page
- "Improving the working speed (performance) of UFT" on the next page
- "Improving the working speed (performance) of UFT" on the next page

Tools in the HP UFT program group

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

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

For Microsoft Windows Vista and Windows Server 2008:

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

For Microsoft Windows 7 and Windows Server 2008 R2:

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

For Microsoft Windows 8:

1. Log in as an administrator.
2. From the Control Panel, select **User Accounts and Family Safety > User Accounts > Change User Account Control Settings**.
3. In the User Account Control Settings window, move the slider to **Never notify**.
4. In the Control Panel, select **System and Security > Administrative Tools > Local Security Policy**.
5. In the Local Security Policy window, in the left pane, select **Local Policies**.
6. In the Local Policies tree, select **Security Options**.
7. In the right pane, select the **User Account Control: Run all administrators in Admin Approval mode** option.
8. Select **Action > Properties** from the menu bar.
9. In the dialog that opens, select **Disabled**.
10. Restart the computer for your changes to take effect.
11. After working with the desired tool, return to the User Account Control Settings window, and restore the slider to its previous position to turn the UAC option on again.
12. Restart the computer for your changes to take effect.

Characters do not display correctly

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

UFT and DEP (Data Execution Prevention)

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

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

Improving the working speed (performance) of UFT

- If you receive a "running low on UFT resources" error message in the Windows system tray, see "Windows Notification Area Messages" on the next page.
- See also "Improving GUI Testing Performance" on page 2258.

Windows Notification Area Messages

Relevant for: GUI tests and components and API testing

UFT sometimes generates system messages in the Windows notification area.

- **UFT resources are running low.** Close any UFT documents that you do not need or remove assets from the solution. If this problem continues, restart UFT.

  This message displays when UFT consumes too much memory.

  To reduce UFT memory usage and improve performance, you can try one or more of the following:

  - Include fewer items in your solution.
  - Open fewer UFT documents simultaneously.
  - Create shorter tests. You can do this by dividing your test into smaller tests, with each test covering fewer scenarios or use cases.
  - Verify that the UFT installation meets all of the minimum system requirements as specified in the *HP Unified Functional Testing Readme*.

In addition, to alleviate the memory shortage, close other unneeded applications that are running on your computer.

If the problem continues, restart UFT.

Troubleshooting and Limitations - Multilingual Applications in GUI Testing

Relevant for: GUI tests and components

This section includes general limitations about multilingual issues in UFT:

- Some user interface items are not localized, for example:
  - Tooltips in the Properties Pane
  - Tooltips and context menu in the editor of the File Content Checkpoint dialog box
  - Compilation information in the Output pane for an API Test
  - Context menu for the search box in the toolbar of the File Content Checkpoint Properties dialog box
  - Error messages when importing a WSDL Web service
  - The default value string for a Date parameter in the Add Input Parameter dialog box for an API test
  - Warning or error messages in the Test Batch Runner, for example, when saving files using the **Save As** command
  - Strings displayed when adding a date-type input parameter in the Properties pane of an API test
  - Some of the UFT user interface related to XML operations
  - XML Validation results in the Run Results Viewer

    Workaround: Install the relevant user interface language .NET Framework Language Pack. You can download it from the Microsoft download center: http://www.microsoft.com/en-us/download/default.aspx

- The image displayed when selecting **View Sample Snapshot** in the UFT Asset Comparison tool is not localized
- UFT may behave unexpectedly when using multi-byte string inputs if it is installed on a VMware operating system.

  Workaround: Set the hardware acceleration of the display driver to **None**. If this does not work, uninstall the VMware display adapter.

- Manually editing scripts in the Editor using the Korean language on Windows XP or Windows 2003 operating systems may cause UFT to malfunction.

  Workaround: Select **Control Panel > Regional and Language Options > Languages** tab **> Details > Advanced** tab and select the **Turn off advanced text services** check box.

  You may need to restart your computer after setting this option.

- There may be cases during a run session when UFT loses the focus of the application you are testing when working in a Korean, Chinese, or Japanese operating system. This may result in a loss of data during the run session.

  Workaround: Run an **Activate** method on the application's window to ensure focus on the window before performing the next step. For example:

  ```
  Window("Notepad").Activate
  ```

- When using Chinese IME to record multiline objects:

  - Some mouse operations on the IME are recorded and are not executed during the run session.

    Workaround: Avoid performing mouse operations on the IME window while recording.

  - Selecting a character from the IME Candidate window is not supported.

- Running the **Type** method on computers with Japanese, Korean, or Chinese operating systems may not work as expected.

  Workaround: Add an English input locale to the computer (using the **Regional Options** or **Regional and Language Options** in your Control Panel).

Additional product-area specific limitations are described in the chapters that describe the relevant functionality.

Troubleshooting and Limitations - Operating Systems in GUI Testing

Relevant for: GUI tests and components

Some operating systems have specific characteristics that affect the normal behavior of UFT functionality. These issues are described in the chapters that describe that functionality. The table below groups these issues by operating system so that you can find all issues that are specific to the operating system you use.

Operating System	Affected Functionality
Windows 7	Recording tests and components
	Using UFT tools from the Windows Start menu
	Exporting run results (in the *HP Run Results Viewer User Guide*)
	Opening and saving tests
	DEP (Data Execution Prevention) - 64-bit only
Windows 8	Accessing in Windows 8 Operating Systems
Windows Server 2008 R2	Recording tests and components
	Exporting data tables
	Exporting run results (in the *HP Run Results Viewer User Guide*)
	Using UFT tools from the Windows Start menu
	DEP (Data Execution Prevention) - see *HP Unified Functional Testing Installation Guide*

Troubleshooting - Naming Conventions

Relevant for: GUI tests and components and API testing

The following table lists the restrictions to consider when naming items in UFT:

Item	Naming Convention
Action (for tests)	• Must be unique in the test. • Cannot begin or end with a space. • Cannot contain the following characters: **\ / : * ? " < > | % ' ! { }**
Action parameter (for tests)	• Case-sensitive. • Must begin with a letter • Cannot contain spaces. • Cannot contain the following characters: **! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ' : " , . / < >**
Application area (for components)	• Cannot exceed 220 characters (including the path). • Cannot contain, begin, or end with spaces. • Cannot contain the following characters: **\ / : " ? < > | * ! { } ' % ; ,** • Cannot contain multibyte punctuation symbols and other multibyte special characters, such as multibyte question marks, multibyte spaces, and multibyte brackets.
Checkpoint	• Cannot begin or end with a space. • Cannot contain **"** (double quotation mark). • Cannot contain the following character combinations: ▪ **:=** ▪ **@@**

<table>
<tr><th>Item</th><th>Naming Convention</th></tr>
<tr><td><b>Component</b>
<b>(for components)</b></td><td>• Cannot exceed 220 characters (including the path).
• Cannot contain, begin, or end with spaces.
• Cannot contain the following characters: <b>\ / : " ? < > | * ! { } ' % ;</b>
• Cannot contain multibyte punctuation symbols and other multibyte special characters, such as multibyte question marks, multibyte spaces, and multibyte brackets.</td></tr>
<tr><td><b>Data Table file (for tests and scripted components)</b></td><td><b>ALM:</b> cannot contain the following characters: <b>! % * { } \ | ' : " / < > ? ; ,</b></td></tr>
<tr><td><b>Data Table > Parameter name (column header)</b>
<b>(for tests and scripted components)</b></td><td>• Must be unique in the sheet.
• Must begin with a letter or underscore.
• Can only contain the following:
■ Letters (excluding special characters, machine-dependent characters, or platform-dependent characters)
■ Numbers
■ Periods
■ Underscores</td></tr>
<tr><td><b>Environment variable (parameter)</b></td><td>• Must begin with a letter.
• Can only contain the following:
■ Letters
■ Numbers
■ Underscores</td></tr>
<tr><td><b>Environment variables file</b></td><td><b>File system:</b> Cannot contain the following characters: <b>! % * { } \ | ' : " / < > ? ;</b>
<b>ALM:</b> Cannot contain the following characters: <b>! % * { } \ | ' : " / < > ? ; ,</b></td></tr>
</table>

Item	Naming Convention
Function library file	**File system:** Cannot contain the following characters: **! % * { } \ | ' : " / < > ? ;** **ALM:** • Cannot contain the following characters: **! % * { } \ | ' : " / < > ? ; ,** • Cannot contain non-English characters
Function / Function argument	• Cannot contain non-English letters or characters. • Function names cannot be the same as a VBScript registered keyword. • Must begin with a letter. • Cannot contain spaces or any of the following characters: **! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ' : "" , / < > ?**
Object repository file	**File system:** Cannot contain the following characters: **! % * { } \ | ' : " / < > ? ;** **ALM:** Cannot contain the following characters: **! % * { } \ | ' : " / < > ? ; ,**
Output value	• Cannot begin or end with a space. • Cannot contain **"** (double quotation mark). • Cannot contain the following character combinations: ▪ **:=** ▪ **@@**
ALM file or folder name	• Cannot exceed 90 characters (including the path). • Cannot begin or end with a space. • Cannot contain the following characters: **\ : * ? " < > | % ' ;** • Cannot contain multibyte punctuation symbols and other multibyte special characters, such as multibyte question marks, multibyte spaces, and multibyte brackets.
Recovery Scenario	**File system:** Cannot contain the following characters: **! % * { } \ | ' : " / < > ? ;** **ALM:** Cannot contain the following characters: **! % * { } \ | ' : " / < > ? ; ,**

Item	Naming Convention
Test name (for tests)	• Cannot exceed 220 characters (including the path). **Note:** When you create a test and store it in the file system, UFT creates a folder with the name you specify, and also a file with the same name inside that folder. The name of this file, including its path, cannot exceed 220 characters. • Cannot begin or end with a space. • Cannot contain the following characters: \ / : * ? " < > | % ' ; • Cannot contain multibyte punctuation symbols and other multibyte special characters, such as multibyte question marks, multibyte spaces, and multibyte brackets.
Test resources (when saving a test with resources)	The path name (resource name and file path together) cannot exceed 256 characters.
Test object class (extensibility only)	• Cannot contain non-English letters or characters. • Cannot contain spaces or any of the following characters: ! @ # $ % ^ & * ( ) + = - [ ] \ { } | ; ' : "" , / < > ?
Test object name	• must be unique within the same class and hierarchy in the object repository • Cannot begin or end with a space. • Cannot contain " (double quotation mark). • Cannot contain the following character combinations: ▪ := ▪ @@
Test object identification properties	• Cannot contain non-English letters or characters. • Cannot contain spaces or any of the following characters: ! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ' : "" , / < > ?
Test object method / Test object method argument	• Cannot contain non-English letters or characters. • Cannot contain spaces or any of the following characters: ! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ' : "" , / < > ?

We appreciate your feedback!

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on User Guide (Unified Functional Testing 12.00)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to sw-doc@hp.com.

Java™
COMPATIBLE