# HP QuickTest Professional for Business Process Testing

Software Version: 9.5

## User's Guide

Manufacturing Part Number: T6513-90035
Document Release Date: January 2008
Software Release Date: January 2008

# Legal Notices

## Warranty

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

The information contained herein is subject to change without notice.

## Restricted Rights Legend

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

## Third-Party Web Sites

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

## Copyright Notices



## Trademark Notices

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

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

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

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

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

Unix® is a registered trademark of The Open Group.

SlickEdit® is a registered trademark of SlickEdit Inc.

## Documentation Updates

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

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

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

**http://ovweb.external.hp.com/lpe/doc_serv/**

# Support

You can visit the HP Software Support Web site at: **www.hp.com/go/hpsoftwaresupport**

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To find more information about access levels, go to:
**http://h20230.www2.hp.com/new_access_levels.jsp**

To register for an HP Passport ID, go to:
**http://h20229.www2.hp.com/passport-registration.html**

# Table of Contents

## PART III: DEFINING FUNCTIONS AND OTHER PROGRAMMING TASKS

## PART IV: WORKING WITH APPLICATION AREAS AND COMPONENTS

## PART VI: RUNNING AND ANALYZING COMPONENTS



## PART VII: MAINTAINING AND DEBUGGING COMPONENTS



## PART VIII: WORKING WITH THE QUICKTEST IDE

## PART X: APPENDIX

# Welcome to This Guide

Welcome to the QuickTest Professional for Business Process Testing User's Guide, which explains how to use QuickTest Professional when working with HP Business Process Testing. This guide describes how to use QuickTest to create and manage the application areas on which components are based, including how to define the various resource files used by components. It also describes how to work with keyword-driven business components and scripted components for Business Process Testing in QuickTest Professional.

Business Process Testing is fully integrated with QuickTest and Quality Center, and is enabled if your license includes Business Process Testing support.

**This chapter includes:**

# How This Guide Is Organized

This guide contains the following parts:

### Part I Introducing Business Process Testing

Provides an overview of QuickTest and the main stages of the testing process when working with Business Process Testing.

### Part II Working with Test Objects and Object Repositories

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

### Part III Defining Functions and Other Programming Tasks

Describes how to enhance your components using function libraries, how to customize the function library window, and how to work with user-defined functions and function libraries in QuickTest.

### Part IV Working with Application Areas and Components

Describes how to create and manage application areas, which include all the resources and settings used by components. This part also describes how to create and work with business components, scripted components, and the Business Component Keyword View, and how to work with checkpoints and output values.

### Part V Configuring Settings

Describes how to modify QuickTest settings to meet your business process testing needs.

### Part VI Running and Analyzing Components

Describes how to run components and their associated function libraries, and how to view and analyze run results.

**Part VII Maintaining and Debugging Components**

Describes how to control run sessions to identify and isolate bugs in your components and function libraries.

**Part VIII Working with the QuickTest IDE**

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

**Part IX Working with Advanced Features**

Describes how to work with recovery scenarios, and how to automate QuickTest operations.

**Part X Appendix**

Provides information on frequently asked questions about QuickTest and describes how to create customized process guidance packages.

## Who Should Read This Guide

This guide is intended for Automation Engineers who are using QuickTest Professional to work with Business Process Testing. Automation Engineers should be experts in automated testing using QuickTest Professional, knowledgeable in keyword-driven testing methodology and processes, and experienced in VBScript programming.

Automation Engineers work together with Subject Matter Experts to create business process tests. Subject Matter Experts use the Business Components module of Quality Center to create business process tests, using resources created by the Automation Engineers. The Business Components module of Quality Center is described in the *HP Business Process Testing User's Guide*.

# QuickTest Professional Online Documentation

QuickTest Professional includes the following online documentation:

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

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

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

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

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

**QuickTest Professional Help** includes:

- **What's New in QuickTest Professional** describes the newest features, enhancements, and supported environments in the latest version of QuickTest.
- **QuickTest User's Guide** describes how to use QuickTest to test your application.
- **QuickTest for Business Process Testing User's Guide** provides step-by-step instructions for using QuickTest to create and manage assets for use with Business Process Testing.
- **QuickTest Professional Add-ins Guide** describes how to work with supported environments using QuickTest add-ins, and provides environment-specific information for each add-in.

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

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

# Additional Online Resources

**Mercury Tours** sample Web site is the basis for many examples in this guide. The URL for this Web site is newtours.demoaut.com.

**Knowledge Base** opens directly to the Knowledge Base landing page on the Mercury Customer Support Web site. Choose **Help** > **Knowledge Base**. The URL for this Web site is support.mercury.com/cgi-bin/portal/CSO/kbBrowse.jsp.

**Customer Support Web site** accesses the HP Software Support Web site. This site enables you to browse the Support Knowledge Base and add your own articles. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. Choose **Help** > **Customer Support Web site**. The URL for this Web site is www.hp.com/go/hpsoftwaresupport.

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

To find more information about access levels, go to:
http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport user ID, go to:
http://h20229.www2.hp.com/passport-registration.html

**Send Feedback** enables you to send online feedback about **QuickTest Professional** to the product team. Choose **Help** > **Send Feedback**.

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

# Typographical Conventions

This guide uses the following typographical conventions:

| | |
|---|---|
| **UI Elements** and **Function Names** | This style indicates the names of interface elements on which you perform actions, file names or paths, and other items that require emphasis. For example, “Click the **Save** button.” It also indicates method or function names. For example, "The **wait_window** statement has the following parameters:" |
| *Arguments* | This style indicates method, property, or function arguments and book titles. For example, “Refer to the *HP User’s Guide*.” |
| <**Replace Value**> | Angle brackets enclose a part of a file path or URL address that should be replaced with an actual value. For example, <**MyProduct installation folder**>**bin**. |
| Example | This style is used for examples and text that is to be typed literally. For example, “Type Hello in the edit box.” |
| CTRL+C | This style indicates keyboard keys. For example, “Press ENTER.” |
| [ ] | Square brackets enclose optional arguments. |
| { } | Curly brackets indicate that one of the enclosed values must be assigned to the current argument. |
| ... | In a line of syntax, an ellipsis indicates that more items of the same format may be included. In a programming example, an ellipsis is used to indicate lines of a program that were intentionally omitted. |
| \| | A vertical bar indicates that one of the options separated by the bar should be selected. |

# Part I

## Introducing Business Process Testing

# 1

# Introduction

Welcome to QuickTest Professional for Business Process Testing. Business Process Testing enables non-technical Subject Matter Experts (working in Quality Center) to collaborate effectively with Automation Engineers (working in QuickTest Professional). Together, you can build, document, and run business process tests, without requiring programming knowledge on the part of the Subject Matter Expert.


**Note:** QuickTest Professional is Unicode compliant, according to Unicode Standard requirements (http://www.unicode.org/standard/standard.html). This enables you to add and update VBScript statements for testing applications developed in many international languages. Unicode represents the required characters using 8-bit or 16-bit code values. You can test non-English language applications, as long as the relevant Windows language support is installed on the computer on which QuickTest Professional is installed (**Start** > **Settings** > **Control Panel** > **Regional Options** or similar).

This guide describes the QuickTest Professional features and options that enable you—the Automation Engineer—to create and modify the automated resources required for Business Process Testing, as well as create components, which are the building blocks of business process tests.

---

**Note:** Although you can also use QuickTest to create scripted components for use in business process tests, this guide focuses on the functionality and features associated primarily with business components. You can find information on the differences between scripted components and business components in QuickTest in Chapter 14, "Creating Scripted Components."

---

**This chapter includes:**

# About Using QuickTest Professional for Business Process Testing

Business Process Testing is a role-based testing model. It enables **Automation Engineers** and **Subject Matter Experts** to work together to test an application's business processes during the application's development life cycle.

Automation Engineers are experts in automated testing. They use QuickTest to define the resources and settings needed to create **components**, which are the building blocks of **business process tests**.

Subject Matter Experts understand the various parts of the application being tested, as well as the business processes that need to be tested, however they may not necessarily have the programming knowledge needed to create automated tests. They use the Business Components and Test Plan modules in Quality Center to create keyword-driven business process tests.

Integration between QuickTest and Quality Center enables the Automation Engineer to effectively create and maintain the required resources and settings, while enabling Subject Matter Experts to create and implement business process tests in a script-free environment, without the need for programming knowledge.

---

**Note:** Each organization defines the roles of Automation Engineer and Subject Matter Expert according to its needs. This guide assumes that you are performing the role of the Automation Engineer as defined above, and that the role of Subject Matter Expert is performed by other personnel in your organization. However, these roles are flexible and depend on the abilities and time resources of the personnel using Business Process Testing. There are no product-specific rules or limitations controlling which roles must be defined in a particular organization, or which types of users can do which Business Process Testing tasks (provided that the users have the correct permissions).

---

# Understanding Business Process Testing

Business Process Testing enables structured testing of an application by combining test automation and automatically generated, easy-to-understand test documentation. Business Process Testing is not dependent on the completion of detailed testing scripts. This enables applications to be tested manually before automated tests are ready. This also enables business process tests to be created and implemented more quickly than other automated tests, enabling potential performance issues to be detected earlier in the development process, before downtime can occur.

**Components** are easily-maintained, reusable units that perform a specific task. They are the building blocks of business process tests. Each component is comprised of several application steps that are logically performed together in a specific order. For example, in a Web application, a login component might be comprised of four steps. Its first step could be to open the application. Its second step could be to enter a user name. Its third step could be to enter a password, and its last step could be to click the **Submit** button on the Web page. By creating and calling functions stored in function libraries, you can enhance the component with additional logic to test important details of the login task.

By design, each component tests a specific part of an application. When combined, components are incorporated into a business process test in a serial flow representing the main tasks performed within a particular business process. For example, a business process test for a flight reservation application may include a login component, a flight finder component, a flight reservation component, a purchasing component, and a logout component. The flight finder, flight reservation, and purchasing components might be reused several times within the same business process test to test multiple reservation scenarios. The test might also include a component that resets the application between flight reservations, enabling the test to perform multiple iterations of flight reservations. The task of creating and running components and business process tests is generally performed by Subject Matter Experts working in Quality Center.

Due to the modularity and reusability of components, they can be used in multiple business process tests. For example, the same login and logout components could be used in conjunction with an analysis (report) component that tests the report and graph generation process in the application, or with a frequent flyer component that tests the business process of subscribing to a frequent flyer program.

QuickTest provides two types of components: **business components** and **scripted components**. Business components (also known as keyword-driven components) are fully integrated with both QuickTest and Quality Center, enabling both you and Subject Matter Experts to create, modify, and run them. Scripted components are more complex components containing programming logic. Due to their complexity, scripted components can be created and modified only in QuickTest. Subject Matter Experts can view scripted components in Quality Center and incorporate them in business process tests, but they cannot modify them.

---

**Note:** Although you can also use QuickTest to create scripted components for use in business process tests, this guide focuses on the functionality and features associated with business components. For information on the differences between scripted components and business components, as well as information on working with scripted components, see Chapter 14, "Creating Scripted Components."

---

Before automated testing resources are available, Subject Matter Experts can define manual steps in the Design Steps tab of each component (using the Quality Center Business Components module). They can add these manual components to a business process test and run the steps manually using the Quality Center Manual Runner. As they define components, Subject Matter Experts can add comments in the Discussion Area of the Details tab (in the Quality Center Business Components module). This enables them to enter any additional information or remarks that they want to communicate to you, the Automation Engineer, such as requests for new operations, future changes planned for the component, or alternative tests in which the component can be used.

During this design phase, you can work with the Subject Matter Experts to define which resources and settings are needed for each component. You can then create individual **application areas** for the various parts of your application based on real testing needs. The application area specifies the settings and resource files used by components when working with business process tests. When a Subject Matter Expert creates a component, the component is always associated with a particular application area, enabling it to access these settings and resource files. After you create the application area and define its settings and resource files, the Subject Matter Expert can incorporate these automated testing resources in business component steps, convert any existing manual components to automated components, and create new automated components.

You can use QuickTest process guidance to guide you through the process of creating a business component. For more information, see "Working with Process Guidance" on page 763.

### Understanding the Application Area

The application area is the foundation upon which components are built. An application area provides a single point of maintenance for all elements associated with the testing of a specific part of an application.

In the application area, you can define specific settings that are relevant for testing a particular part of your application. For example, you can define settings that instruct QuickTest to load specific add-ins at the start of a run session, run a component only on specified applications, activate a recovery scenario under particular conditions, and so forth. You can also specify the keywords that are available to any component that is associated with that particular application area.

An important aspect of application areas are the resource files that can be used by a component. After you create these resource files you store them in the same Quality Center project used by the Subject Matter Experts who create and run the business process tests for the specific application. Typical resource files include function libraries and shared object repositories.

You create, populate, and maintain shared object repository files that are used by QuickTest to identify the objects in your application. You define and modify test object information in shared object repositories using the QuickTest Object Repository Manager. After you associate shared object repository files with the application area, you can prioritize them according to relevance. By associating a shared object repository with an application area, any component based on that application area will have access to all of its test objects and other elements. For more information, see Chapter 4, "Working with Objects," and Chapter 6, "Managing Object Repositories."

You also create function libraries that contain functions, or **operations** (also known as keywords), that can be called by a component. These functions contain programming logic that encapsulates the steps needed to perform a particular task, and they enhance the functionality of the component that calls them. You can use QuickTest to create these function libraries. You can also use the QuickTest Function Definition Generator to insert basic function definitions, and then complete each function by adding its code.

After you associate function library files with an application area, you can prioritize them according to relevance. By associating a function library with an application area, any component based on that application area will have access to all public functions defined within that function library. For more information on working with function libraries, see Chapter 11, "Working with User-Defined Functions and Function Libraries."

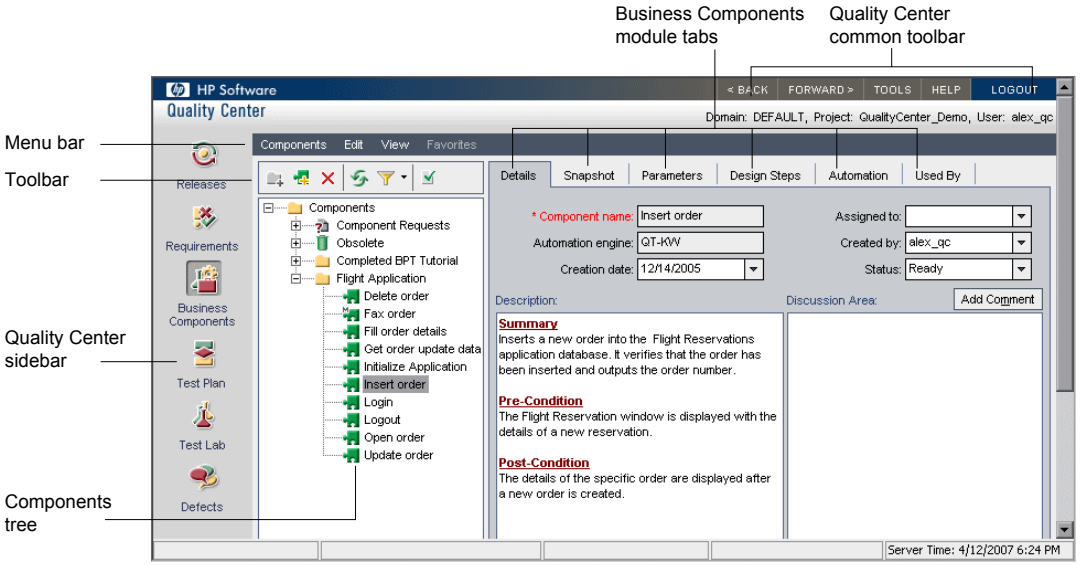
You can create multiple application areas—each one focusing on a particular part (area) of the application being tested. For example, for a flight reservation application, one application area could be created for the login module, another application area for the flight search module, another for the flight reservation module, and still another for the billing module. For more information on application areas, see Chapter 12, "Working with Application Areas."

In addition to creating and maintaining the resource files associated with the application areas, you can also use QuickTest to debug components and their associated function libraries. You can also create components in QuickTest, although this is more often done by Subject Matter Experts using Quality Center. For more information, see Chapter 13, "Working with Business Components."

You can use QuickTest process guidance to guide you through the process of creating an application area. For more information, see "Working with Process Guidance" on page 763.

## Creating Components in the Quality Center Business Components Module

The Subject Matter Expert creates new components and defines them in the Quality Center Business Components module.




The Business Component module includes the following:

- **Details tab.** Provides a general summary of the component's purpose or goals, and the condition of the application before and after a component is run (its pre-conditions and post-conditions).
- **Snapshot tab.** Displays an image that provides a visual cue or description of the component's purpose or operations.
- **Parameters tab.** Specifies the input and output component parameter values for the business component. Implementing and using parameters enables a component to receive data from an external source and to pass data to other components in the business process test flow.

- **Design Steps tab.** Enables you to create or view the manual steps of your business component, and to automate it if required.
- **Automation tab.** Displays or provides access to automated components. For keyword-driven components, enables you to create and modify the steps of your automated business component in a keyword-driven, table format, and provides a plain-language textual description of each step of the implemented component.
- **Used by tab.** Provides details about the business process tests that include the currently selected component. The tab also includes a link to the relevant business process test in the Test Plan module.

## Creating Business Process Tests in the Quality Center Test Plan Module

To create a business process test, the Subject Matter Expert selects (drags and drops) the components that apply to the business process test and configures their run settings.

---

**Note:** When you run a business process test from Quality Center, the test run may also be influenced by settings in the QuickTest Remote Agent. For more information on the QuickTest Remote Agent, refer to the *QuickTest Professional User's Guide*.

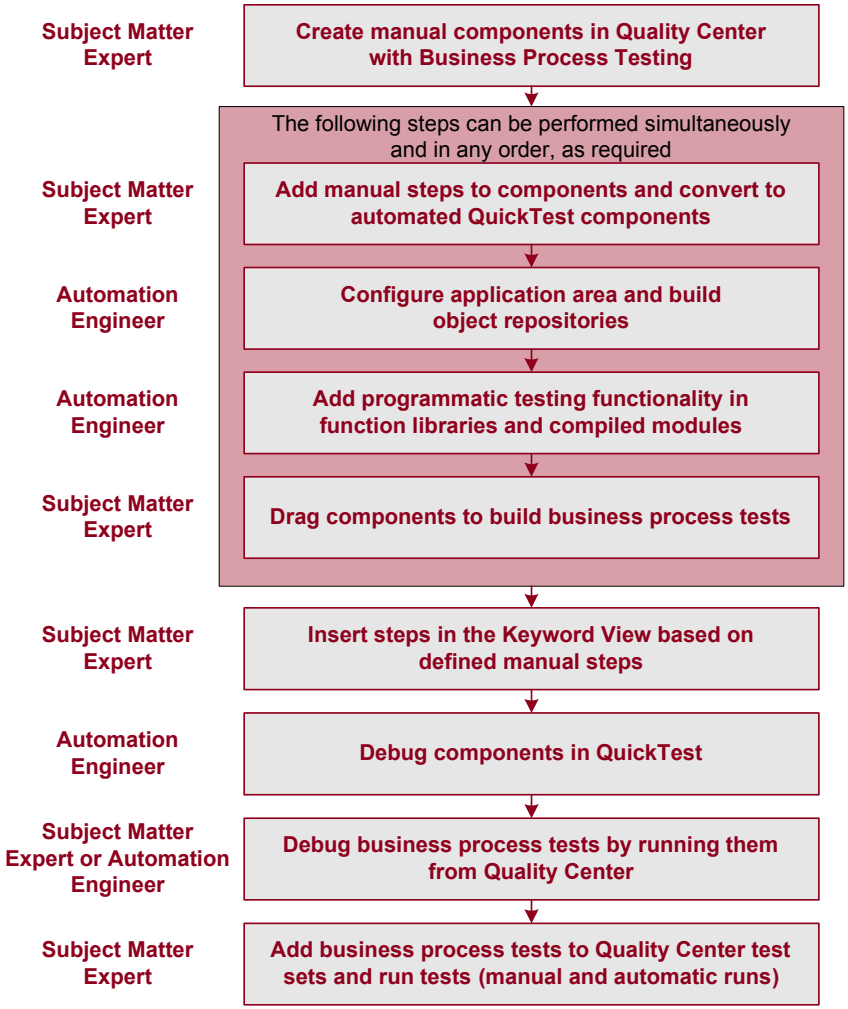
---

Each component can be used differently by different business process tests. For example, in each test the component can be configured to use different input parameter values or run a different number of iterations.

If, while creating a business process test, the Subject Matter Expert realizes that a component has not been defined for an element that is necessary for the business process test, the Subject Matter Expert can submit a component request from the Test Plan module.

## Understanding the Business Process Testing Workflow

The Business Process Testing workflow may differ according to your testing needs. Following is an example of a common workflow:



## Understanding QuickTest Professional for Business Process Testing Terminology

The following terminology, specific to QuickTest Professional for Business Process Testing, is used in this guide:

**Application Area.** A collection of resources and settings that are used for the creation and implementation of business components. These include function libraries, shared object repositories, keywords, testing preferences, and other testing resources, such as recovery scenarios. An application area provides a single point of maintenance for all elements associated with the testing of a specific part of your application. You can define separate application areas for each part of your application and then associate your components with the appropriate application areas.

**Business Component** (or **Component**)**.** An easily-maintained, reusable unit comprising one or more steps that perform a specific task. Business components may require input values from an external source or from other components, and they can return output values to other components.

Also known as Keyword-Driven Component.

**Manual Component.** A non-automated business component created in Quality Center. In QuickTest, you can view and work with manual components only after converting them to automated business components.

**Scripted Component.** An automated component that can contain programming logic and can be edited in QuickTest using the Keyword View, the Expert View, and other QuickTest tools and options.

**Keyword View.** A spreadsheet-like view that enables tests and components to be created, viewed, and debugged using a keyword-driven, modular, table format.

**Function Library.** A document containing VBScript functions, subroutines, modules, and so forth. These functions can be used as operations (keywords) in components. You can create and debug function library documents using the QuickTest function library editor.

**Business Process Test.** A scenario comprising a serial flow of business components, designed to test a specific business process of an application.

**Component Input Parameters.** Variable values that a business component can receive and use as the values for specific, parameterized steps in the component.

**Component Output Parameters.** Values that a business component can return. These values can be viewed in the business process test results and can also be used as input for a component that is used later in the test.

**Local Input Parameters.** Variable values defined within a component. These values can be received and used by a later parameterized step in the same component.

**Local Output Parameters.** Values that an operation or a component step can return for use within the same component. These values can be viewed in the business process test results and can also be used as input for a later step in the component.

**Roles.** The various types of users who are involved in Business Process Testing.

**Automation Engineer.** An expert in QuickTest Professional automated testing. The Automation Engineer defines and manages the resources that are needed to create and work with business components. The Automation Engineer creates application areas that specify all of the resources and settings needed to enable Subject Matter Experts to create business components and business process tests in Quality Center. The Automation Engineer can create and modify function libraries, and populate a shared object repository with test objects that represent the different objects in the application being tested. The Automation Engineer can also create and debug business components in QuickTest.

**Subject Matter Expert.** A person who has specific knowledge of the application logic, a high-level understanding of the entire system, and a detailed understanding of the individual elements and tasks that are fundamental to the application being tested. The Subject Matter Expert uses Quality Center to create and run components and business process tests.

# Setting Required Access Permissions

You must make sure the following access permissions are set in order to run QuickTest Professional.

## Permissions Required to Run QuickTest Professional

You must have the following file system permissions:

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

You must have the following registry key permissions:

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

## Permissions Required When Working with Quality Center

You must have the following Quality Center permissions:

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

### Permissions Required When Working with Business Process Testing

The Quality Center 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 Quality Center Project Administrator can assign permissions for adding, modifying, and deleting folders, components, steps, and parameters in the Business Components module of a Quality Center project.

---

**Note:** To modify application areas, you must have the required permissions 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.

---

You need to make sure you have the required Quality Center permissions before working with business components and application areas. For more information on setting user group permissions in the Business Components module, refer to the *Business Process Testing User's Guide*.

## Using the Sample Site

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

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

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

## Modifying License Information

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

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

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

For information on modifying your license information, refer to the *QuickTest Professional Installation Guide*.

## Updating QuickTest Software

By default, QuickTest automatically checks for online software updates each time you start the application. You can also manually check for updates at any time by choosing **Help** > **Check for Updates** from within QuickTest, or by choosing **Start** > **Programs** > **QuickTest Professional** > **Check for Updates**.

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

---

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

---

# 2

# QuickTest at a Glance

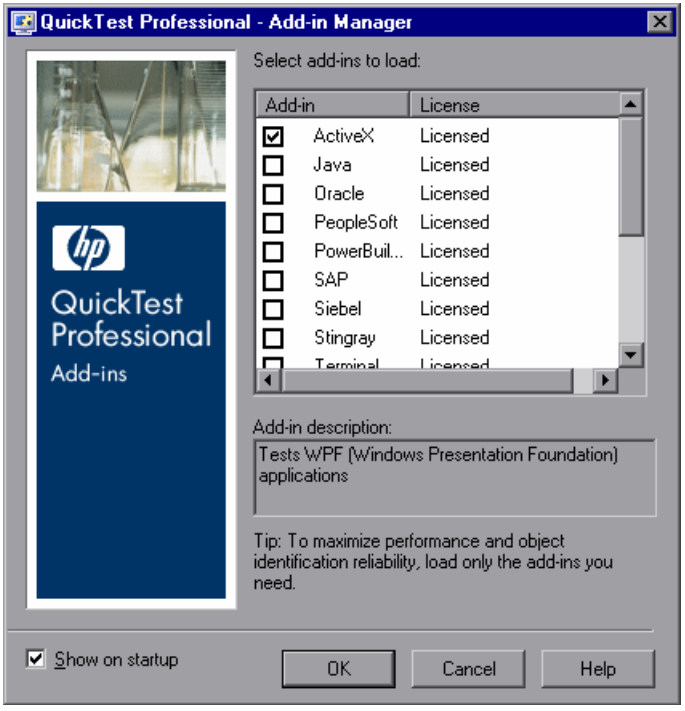
This chapter explains how to start QuickTest and introduces the QuickTest window.

**This chapter includes:**

# Starting QuickTest

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

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




---

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

---

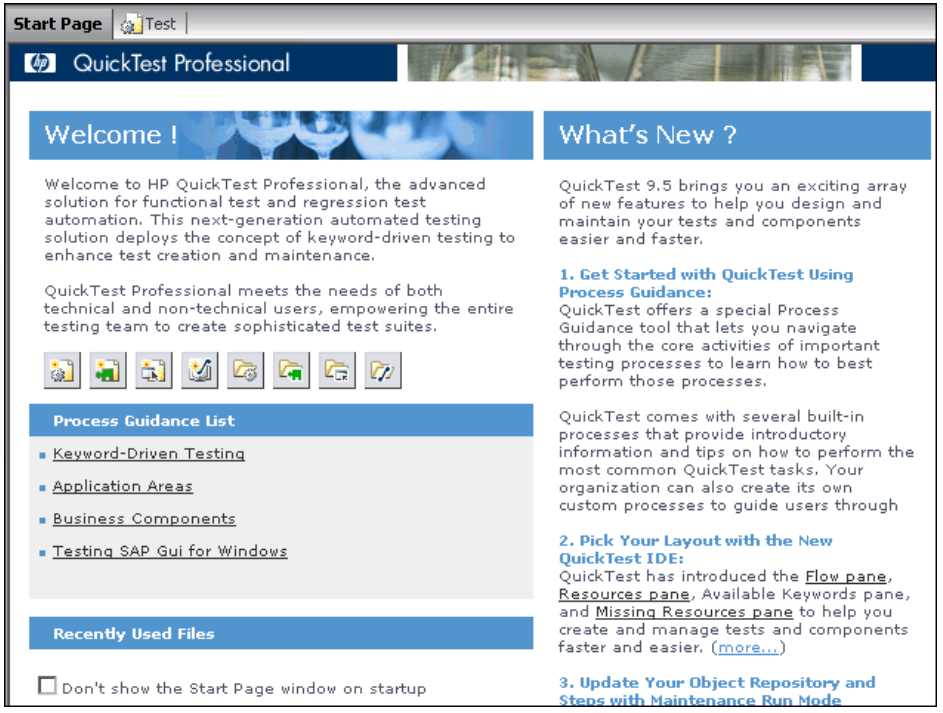
---

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

---

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

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



In the Start page, you can:

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

For more information on the Start Page, see "Start Page" on page 59.

## Connecting to Your Quality Center Project

To work with business process testing, you must connect QuickTest to the Quality Center server on which your Quality Center project is stored. This server handles the connections between QuickTest and your Quality Center project. For Business Process Testing, you can connect from QuickTest Professional 9.0 or later only to Quality Center 9.0 or later.

Your Quality Center project stores component and run session information for the application you are testing, including all of the resource files and settings needed to create and run business process tests. The first time you connect QuickTest to a Quality Center server and project, QuickTest sets up default Business Process Testing folders and files in your project. This enables you to prepare the resources and settings needed for business components, as well as create, work with, and debug business components using the intuitive, keyword-driven Keyword View.

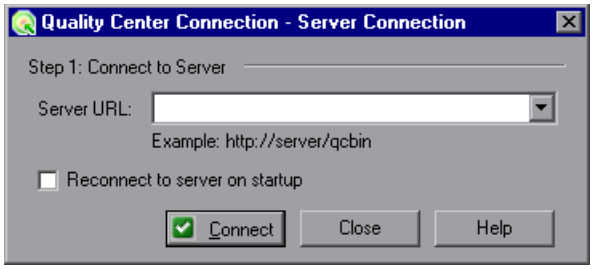
---

**Note:** Quality Center projects are password protected, so you must provide a user name and a password.

---

**To connect QuickTest to a Quality Center server:**

**1** Choose **File** > **Quality Center Connection** or click the **Quality Center Connection** toolbar button. The Quality Center Connection - Server Connection dialog box opens.




**2** In the **Server URL** box, type the URL address of the Web server where Quality Center is installed.

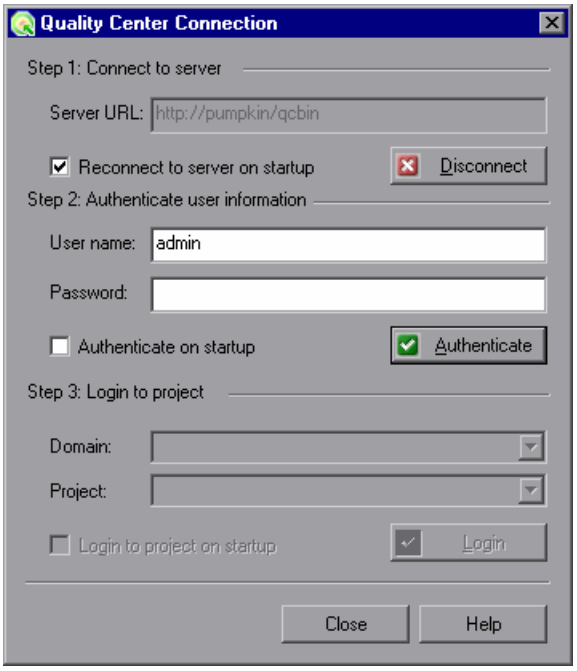
---

**Note:** You can choose a Quality Center server accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).

---

**3** To automatically reconnect to the Quality Center server the next time you open QuickTest, select the **Reconnect to server on startup** check box.

**4** Click **Connect**. The Quality Center Connection dialog box opens.




The Quality Center server name is displayed in read-only format in the Server URL box.

**5** In the **User name** box, type your Quality Center user name.

**6** In the **Password** box, type your Quality Center password.

**7** Click **Authenticate** to authenticate your user information against the Quality Center server.

After your user information has been authenticated, the fields in the **Authenticate user information** area are displayed in read-only format. The **Authenticate** button changes to a **Change User** button.

---

**Tip:** You can log in to the same Quality Center server using a different user name by clicking **Change User**, and then entering a new user name and password and clicking **Authenticate** again.

---

**8** In the **Domain** box, select the domain that contains the Quality Center project. Only those domains that you have permission to connect to are displayed.

**9** In the **Project** box, select the project with which you want to work. Only those projects that you have permission to connect to are displayed.

**10** Click **Login**.

**11** To automatically reconnect to the Quality Center server the next time you open QuickTest, select the **Reconnect to server on startup** check box.

**12** If the **Reconnect to server on startup** check box is selected, then the **Authenticate on startup** check box is enabled. To automatically authenticate your user information the next time you open QuickTest, select the **Authenticate on startup** check box.

**13** If the **Authenticate on startup** check box is selected, the **Login to project on startup** check box is enabled. To log in to the selected project on startup, select the **Login to project on startup** check box.

---

**Note:** The first time you connect to a Quality Center server, QuickTest sets up default Business Process Testing folders and files in your Quality Center project.

---

**14** Click **Close** to close the Quality Center Connection dialog box. The Quality Center icon is displayed on the status bar to indicate that QuickTest is currently connected to a Quality Center project.


---

**Tip:** To view the current Quality Center connection, point to the **Quality Center** icon on the status bar. A tooltip displays the Quality Center server name and project to which QuickTest is connected. To open the Quality Center Connection dialog box, double-click the **Quality Center** icon.

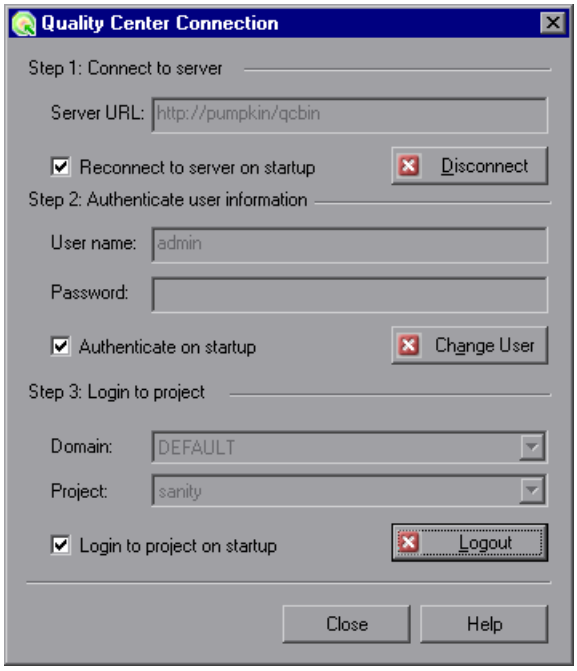
---

## Disconnecting QuickTest from Quality Center

You can disconnect QuickTest from a Quality Center project or from a Quality Center server at any time. Note that if you disconnect QuickTest from a Quality Center server without first disconnecting from a project, the QuickTest connection to that project database is automatically disconnected. However, do not disconnect QuickTest from Quality Center while a QuickTest component, application area, or shared resource (such as a shared object repository) is opened from Quality Center, or while QuickTest is using a shared resource from Quality Center.

**To disconnect QuickTest from Quality Center:**

**1** Choose **File** > **Quality Center Connection** or click the **Quality Center Connection** toolbar button. The Quality Center Connection dialog box opens.




**2** To disconnect QuickTest from the selected project, in the **Step 3: Login to project** area, click **Logout**.

**3** To disconnect QuickTest from the selected Quality Center server, in the **Step 1: Connect to server** area, click **Disconnect**.

---

**Tip:** You can log in to the same Quality Center server using a different user name by clicking **Change User** and then entering a new user name and password and clicking **Authenticate** again.

---

**4** Click **Close** to close the Quality Center Connection dialog box.

# QuickTest Window

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

You can work on one component or application area and one or more function libraries simultaneously. (For your convenience, you can display one active document in the document area, or you can cascade or tile your open documents.) For more information, see "Working With Multiple Documents" on page 738.

## Document Area

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

- **Business Component.** Enables you to create, view, and modify your business component using keywords and operations. For more information, see Chapter 15, "Working with the Keyword View."
- **Scripted Component.** Enables you to create, view, and modify your scripted component in Keyword View or Expert View (described below). For more information on scripted components, see Chapter 14, "Creating Scripted Components." For more information on the Expert View, see the *HP QuickTest Professional User's Guide*.
- **Application Area.** Enables you to define resources and settings for your components. For more information, see Chapter 12, "Working with Application Areas."
- **Function Library.** Enables you to create, view, and modify functions (operations) for use with your component. For more information, see Chapter 11, "Working with User-Defined Functions and Function Libraries."
- **Start Page.** Welcomes you to QuickTest and provides links to Process Guidance. You can use the shortcut buttons to open new and existing documents. For more information, see "Start Page" on page 59.

### Key Elements in the QuickTest Window

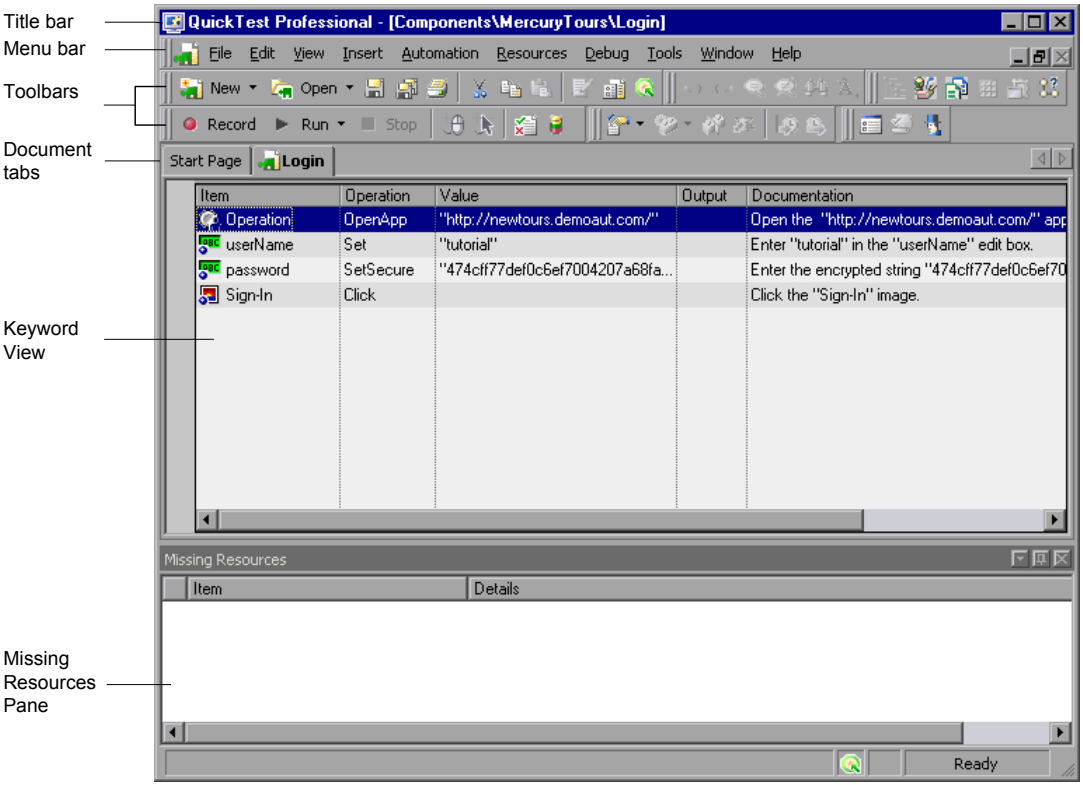
In addition to the document area, the QuickTest window contains the following key elements:

- **QuickTest title bar.** Displays the name of the active document. If changes have been made since it was last saved, an asterisk (*) is displayed next to the document name in the title bar.
- **Menu bar.** Displays menus of QuickTest commands.
- **Standard toolbar.** Contains buttons to assist you in managing your document.
- **Automation toolbar.** Contains buttons to assist you in the testing process.
- **Debug toolbar.** Contains buttons to assist you in debugging your document. (Not displayed by default)
- **Edit toolbar.** Contains buttons to assist you in editing your function library.
- **Insert toolbar.** Contains buttons to assist you when working with statements in your function library.
- **Tools toolbar.** Contains buttons with tools to assist you in the testing process.
- **View toolbar.** Contains buttons to assist you in viewing your document.
- **Action toolbar.** Contains buttons and a list of actions, enabling you to view the details of an individual action or the entire test flow. (Not displayed by default)
- **Document tabs and scroll arrows.** Enables you to navigate open documents in the document area by selecting the tab of the document you want to activate (bring into focus). When there is not enough space in the document area to display all of the tabs simultaneously, you can use the left and right arrows to scroll between your open documents.
- **Keyword View.** Contains each step, in a modular, icon-based table. For more information, see Chapter 15, "Working with the Keyword View."
- **Status bar.** Displays the status of the QuickTest application and other relevant information.

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

- **Debug Viewer pane.** Assists you in debugging your document. The Debug Viewer pane contains the **Watch**, **Variables**, and **Command** tabs.
- **Information pane.** Displays a list of syntax errors found in your function library scripts.
- **Missing Resources pane.** Provides a list of the resources that are specified in your component but cannot be found, such as unmapped shared object repositories and parameters that are connected to shared object repositories. The Missing Resources pane then enables you to locate or remove them from your application area.
- **Process Guidance panes.** Displays two panes that provide procedures and descriptions on how to best perform specific processes, such as creating a component in QuickTest. The Process Guidance Activities pane lists the activities that you can perform, such as adding steps to a component. The Process Guidance Description pane describes the tasks that you need to perform for a selected activity. Your organization may also provide you with process guidance that is accessible from these panes.
- **Available Keywords pane.** Displays all the keywords available to your component. Enables you to drag and drop objects or calls to functions into your component.
- **Resources pane.** Displays all the resources associated with your current component and enables you to view and open these resources.

You can customize the layout of the QuickTest window by moving, resizing, displaying, or hiding most of the elements. QuickTest remembers your preferred layout settings and opens subsequent sessions with your customized layout. For more information, see “Modifying the QuickTest Window Layout” on page 729.



## Changing the Appearance of the QuickTest Window

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

**To change the appearance of the main QuickTest window:**

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

---

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

---

---

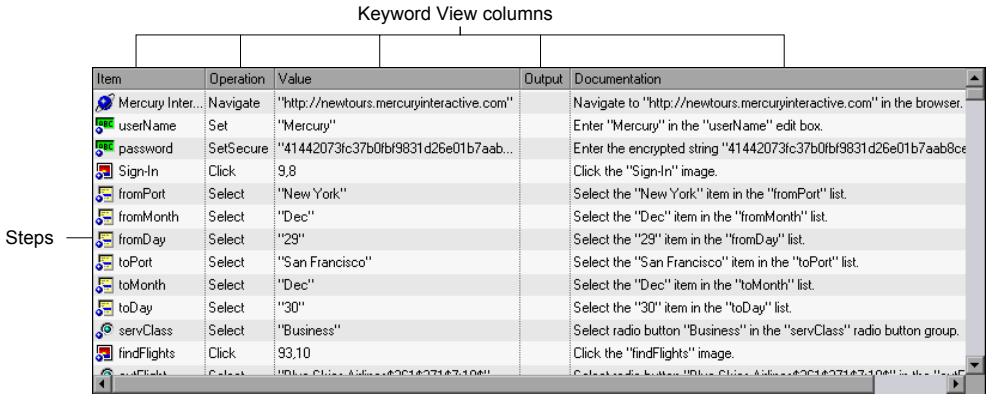
**Tip:** You can also change the theme used for the Test Results window. For more information, see "Changing the Appearance of the Test Results Window" on page 632.

---

# Keyword View

The Keyword View enables you to create and view the steps of your 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 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.



# Application Area

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), associated add-ins, and recovery scenario files. You define these assets in the application area window.

The application area window contains four panes that are accessed by the buttons in the left sidebar:

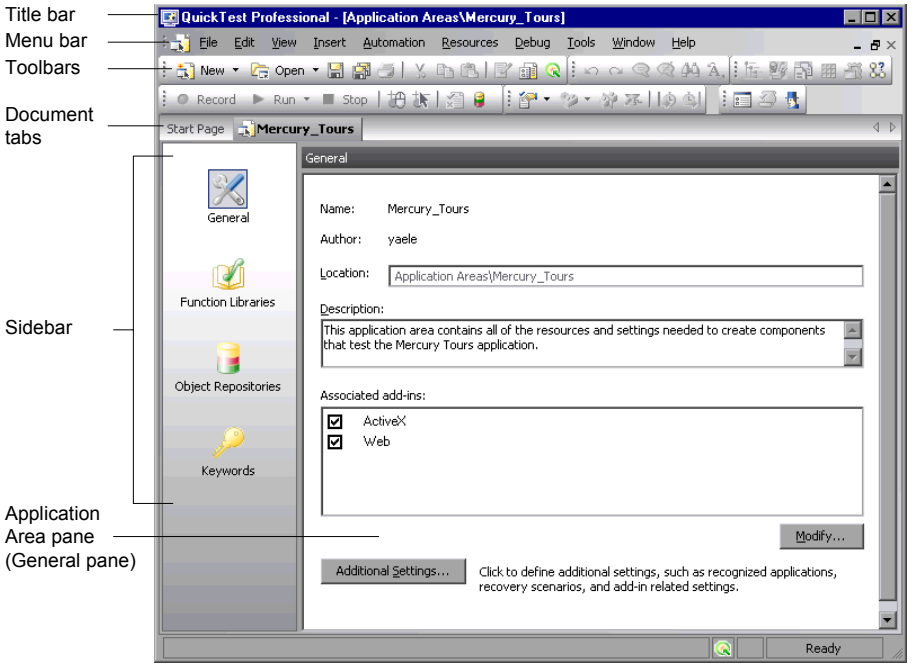
**General.** Displays general information about the application area and enables you to modify its general settings, such as specifying associated add-ins, recovery scenarios, and other settings.

**Function Libraries.** Enables you to associate function libraries with this application area and to prioritize them.

**Object Repositories.** Enables you to associate shared object repositories with this application area and to prioritize them.

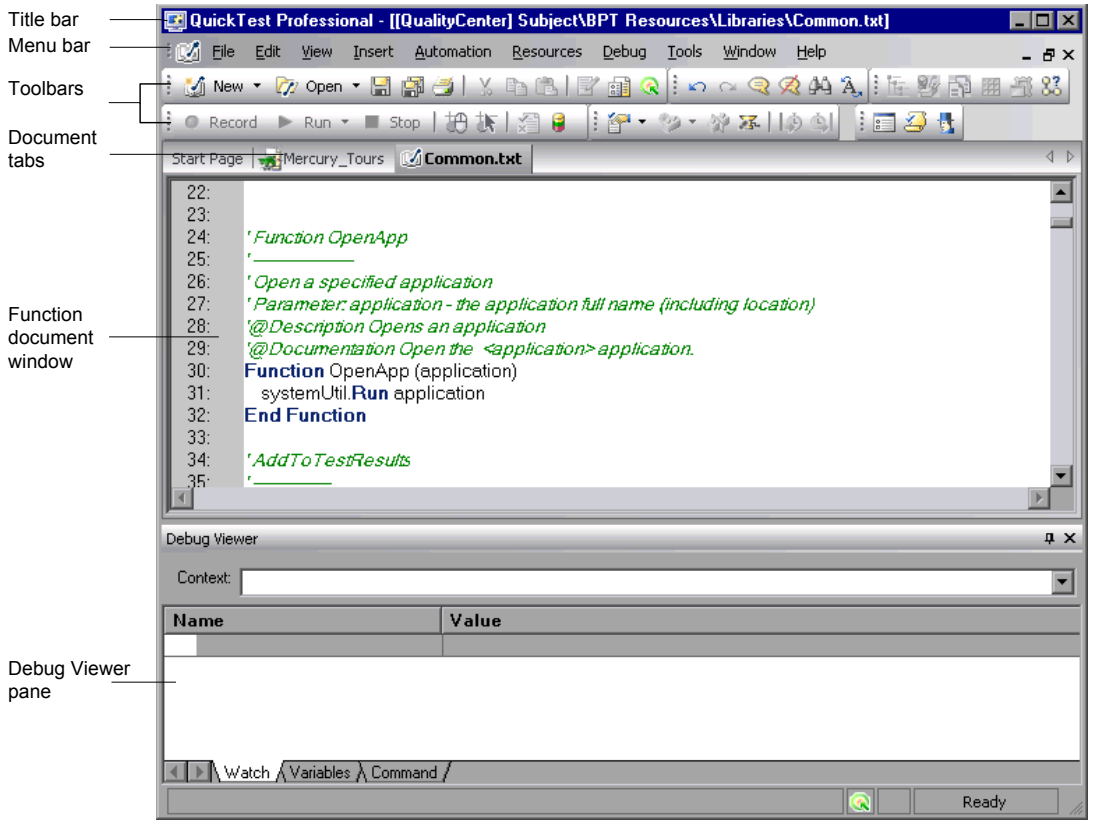
**Keywords.** Enables you to set the keywords that are available to this application area and to view their individual properties.

For more information, see Chapter 12, “Working with Application Areas.”



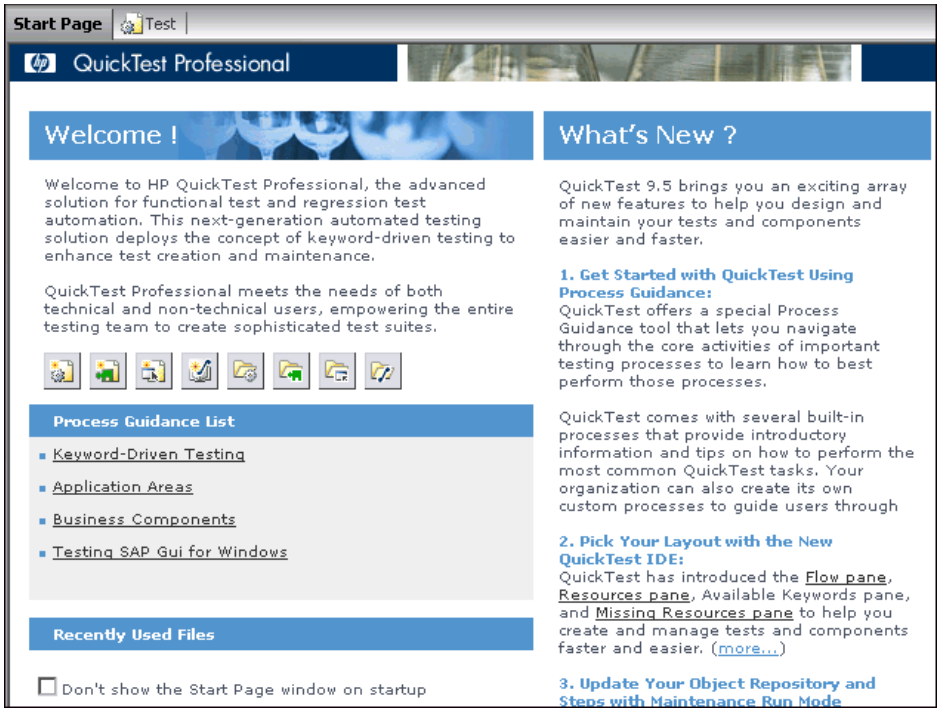
# Function Library

QuickTest provides a built-in editor that enables you to create and debug function libraries using the same editing features that are available in the Expert View. Each function library is a separate QuickTest document containing VBscript functions, subroutines, classes, modules, and so forth. Each function library opens in its own window, in addition to the component that is already open. You can work on one or several function libraries at the same time. After you finish editing a function library, you can close it, leaving your QuickTest session open. You can also close all open function libraries simultaneously. For more information, see Chapter 11, "Working with User-Defined Functions and Function Libraries."



# Start Page

The Start Page welcomes you to QuickTest and describes the new features in this release—including links to more information about these features. It also provides links to Process Guidance, a tool that offers best practices for working with QuickTest. If your organization has descriptions for its own custom processes, these processes may also be available from the **Process Guidance List**. (For more information, see "Working with Process Guidance" on page 763.)



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

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

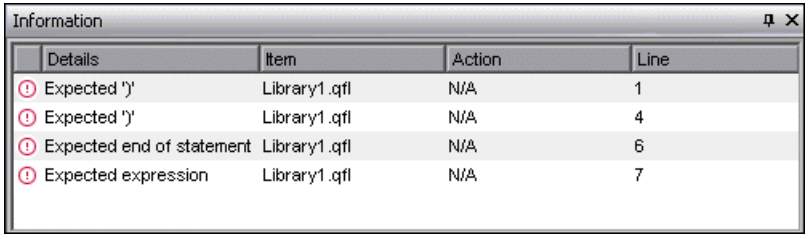
---

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

---

# Information Pane

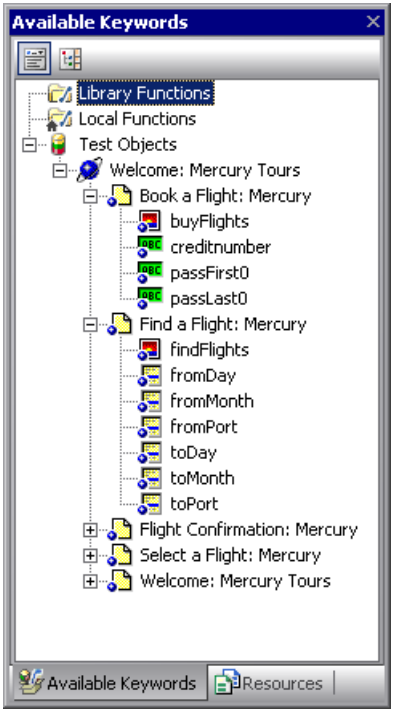
The Information pane provides a list of syntax errors in your function library scripts. To show or hide the Information pane, choose **View** > **Information** or click the **Information** button.




You can double-click a syntax error to locate the error in the function library, and then correct it. For more information, see “Handling VBScript Syntax Errors” on page 330.

# Available Keywords Pane

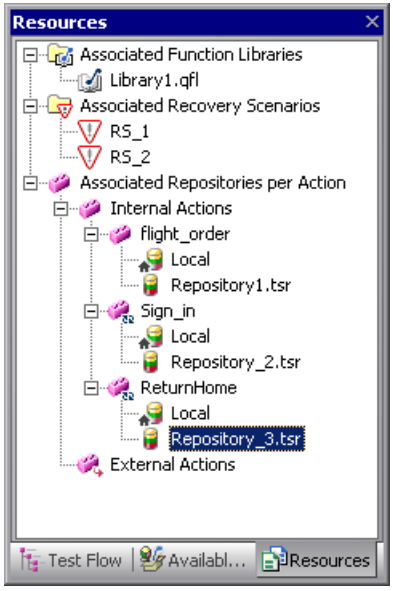
The Available Keywords pane enables you to drag and drop objects or calls to functions into your component. When you drag and drop an object into your component, QuickTest inserts a step with the default operation for that object. When you drag and drop a function into your component, QuickTest inserts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




For more information, see “Understanding the Available Keywords Pane” on page 755.

# Resources Pane

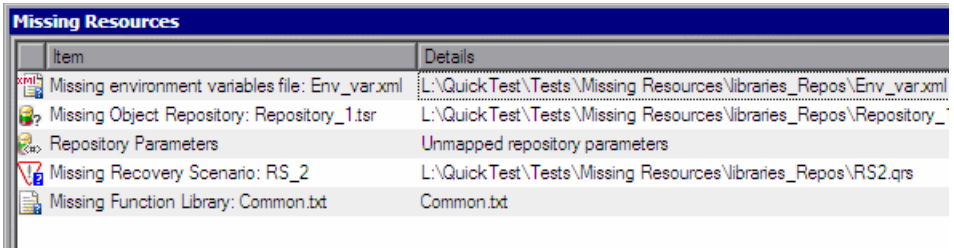
Components are associated with resources such as function libraries, recovery scenarios, and object repositories (via an application area). QuickTest displays all the resources associated with a component in the Resources pane. The Resources pane enables you to view and open all of the resources in your component. To view the Resources pane, click the **Resources Pane** button or choose **View** > **Resources**.




For more information, see “Understanding the Resources Pane” on page 759.

# Missing Resources Pane

The Missing Resources pane provides a list of the resources that are specified in your test but cannot be found. Missing resources can include missing function libraries, missing recovery scenarios, a missing XML file used to store environment variables, unmapped shared object repositories, and parameters that are connected to shared object repositories. To show or hide the Missing Resources pane, choose **View** > **Missing Resources** or click the **Missing Resource** button.




Each time you open your function library, QuickTest automatically checks that all specified resources are accessible. If it finds any resources that are not accessible, QuickTest lists them in the Missing Resources pane. If the Missing Resources pane is not currently displayed, QuickTest automatically opens it when a missing resource is detected.

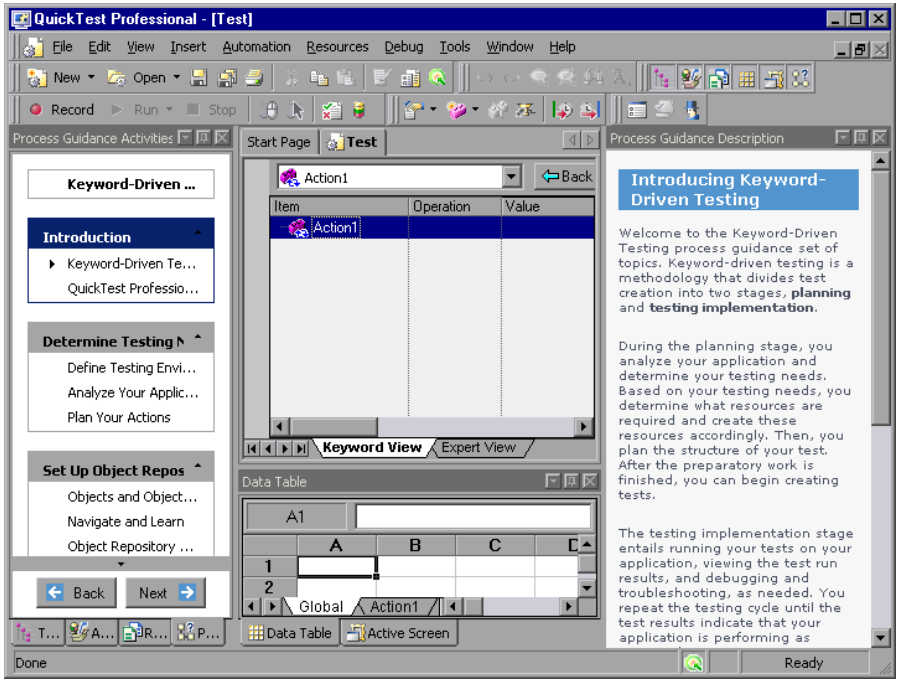
You can double-click a missing resource to remap it or remove it. You can also filter the pane to display a specific type of missing resource, such as Missing Object Repository and hide the other types.

For more information, see “Handling Missing Resources” on page 741.

# Process Guidance Panes

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

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




The **Process Guidance Activities** pane (shown on the left) lists the activities that are part of the selected process. The **Process Guidance Description** pane (shown on the right) displays the topic (description), for the selected activity. For more information, see Chapter 31, “Working with Process Guidance.”

# Debug Viewer Pane

The Debug Viewer pane contains three tabs to assist you in debugging your function library—Watch, Variables, and Command. To view the Debug Viewer pane, choose **View** > **Debug Viewer**.

### Watch

The Watch tab enables you to view the current value of any variable or VBScript expression that you added to the Watch tab.

### Variables

During a run session, the Variables tab displays the current value of all variables that have been recognized up to the last step performed in the run session.

### Command

The Command tab enables you to run a line of script to set or modify the current value of a variable or VBScript object in your function library. When you continue the run session, QuickTest uses the new value that was set in the command.

For more information on using the Debug Viewer pane, see Chapter 25, "Debugging Components and Function Libraries."

# Using QuickTest Commands

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

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


- "Clicking Commands on a Toolbar" on page 68
- "File Menu Commands" on page 71
- "Edit Menu Commands" on page 74
- "View Menu Commands" on page 77
- "Insert Menu Commands" on page 78
- "Automation Menu Commands" on page 79
- "Resources Menu Commands" on page 80
- "Debug Menu Commands" on page 81
- "Tools Menu Commands" on page 83
- "Window Menu Commands" on page 84
- "Help Menu Commands" on page 85
- "Data Table Menu Commands" on page 87
- "Other QuickTest Commands" on page 88

## Clicking Commands on a Toolbar

You can perform some QuickTest commands by clicking buttons on the toolbars. QuickTest has the following built-in toolbars—the **Standard** toolbar, the **Edit** toolbar, the **Automation** toolbar, the **View** toolbar, the **Insert** toolbar, the **Tools** toolbar, and the **Debug** toolbar, and the **Action** toolbar.

---

**Notes:**

- Not all toolbars are relevant for all document types. Only those toolbars that are relevant for components, application areas, and function libraries are described here.
- You can display, hide, or move the toolbars, but you cannot customize them.

---

### Standard Toolbar

The **Standard** toolbar contains buttons for managing a component, application area, or function library.


For information on the **Standard** toolbar buttons, see "File Menu Commands" on page 71 and "Resources Menu Commands" on page 80.

---

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

---

For more information on working with business components, see Chapter 13, "Working with Business Components." For more information on working with scripted components, see Chapter 14, "Creating Scripted Components." For more information on working with application areas, see Chapter 12, "Working with Application Areas." For more information on working with function libraries, see Chapter 11, "Working with User-Defined Functions and Function Libraries."

### Automation Toolbar

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


For information on the **Automation** toolbar buttons, see "Automation Menu Commands" on page 79.

### Debug Toolbar

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

For information on the **Debug** toolbar buttons, see "Insert Menu Commands" on page 78.

### Edit Toolbar

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

For information on the **Edit** toolbar buttons, see "Edit Menu Commands" on page 74.

### Insert Toolbar

The **Insert** toolbar contains buttons for the commands used when working with function libraries.

For information on the **Insert** toolbar buttons, see "Insert Menu Commands" on page 78.

---

**Note:** Only the **Step Generator** and **Function Definition Generator** buttons are relevant for function libraries. None of these buttons are relevant for components.

---

### Tools Toolbar

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

For information on the **Tools** toolbar buttons, see "Tools Menu Commands" on page 83.

The **Check Syntax** button is relevant for only for function libraries (and QuickTest tests), and not for components.

### View Toolbar

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

For information on the **View** toolbar buttons, see "View Menu Commands" on page 77.

**Note:** The **Active Screen** and **Data Table** buttons apply only to tests.

## Performing QuickTest Commands

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

## File Menu Commands

You can manage your component, application area, or function library using the following **File** menu commands:

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **New > Test** | CTRL+N | Creates a new test. |
|  | **New > Business Component** | CTRL+SHIFT+N | Creates a new business component. |
|  | **New > Scripted Component** | | Creates a new scripted component. |
|  | **New > Application Area** | CTRL+ALT+N | Creates a new application area. |
|  | **New > Function Library** | SHIFT+ALT+N | Creates a new function library. |
|  | **Open > Test** | CTRL+O | Opens an existing test. |
|  | **Open > Business/Scripted Component** | CTRL+SHIFT+O | Opens an existing business or scripted component. |
|  | **Open > Application Area** | CTRL+ALT+O | Opens an existing application area. |
|  | **Open > Function Library** | SHIFT+ALT+O | Opens an existing function library. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
| | **Close** | | Closes the active function library. |
| | **Close All Function Libraries** | | Closes all open function libraries. |
|  | **Quality Center Connection** | | Opens the Quality Center Connection dialog box, enabling you to connect to a Quality Center project.<br>**Tip:** Double-click the Quality Center icon on the status bar to manage your connection.<br>Point to the Quality Center icon on the status bar to view connection information.<br>  |
|  | **Save** | CTRL+S | Saves the active document. |
| | **Save As** | | Opens the relevant Save dialog box so you can save the open document. |
|  | **Save All** | | Saves all open documents. |
|  | **Enable Editing** | | Makes read-only function libraries editable. |
| | **Export Test to Zip File** | CTRL+ALT+S | Creates a zip file of the active document. |
| | **Import Test from Zip File** | CTRL+ALT+I | Imports a document from a zip file. |
| | **Convert to Scripted Component** | CTRL+ALT+C | Converts a business component to a scripted component. |
|  | **Print** | CTRL+P | Prints the active document. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
| | **Print Preview** | | Displays the Keyword View as it will look when printed and enables you to modify the page setup. |
| | **Process Guidance Management** | | Opens the Process Guidance Management dialog box, enabling you to manage the list of processes that are available in QuickTest. |
|  | **Settings** | | Opens the Settings dialog box, enabling you to define settings for the open document. (Not relevant for function libraries) |
| | **Change Application Area** | | Enables you to associate the component with a different application area. |
| | **Associate Library '<Function Library Name>' with '<Document Name>'** | | Associates the active function library with the open document. (Available only from function libraries) |
| | **Recent Files** | | Lists the recently viewed files. |
| | **Exit** | | Closes the QuickTest session. |

Many of the **File** menu commands are also available from the **Standard** toolbar (described on page 68).

## Edit Menu Commands

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

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
| | **Action > Action Properties** | | Enables you to specify options, parameters, and associated object repositories for a stored action. |
| | **Action > Action Call Properties** | | Enables you to specify the number of run iterations according to the number of rows in the Data Table, and to define the values of input parameters and the storage location of output parameters. |
| | **Step Properties > Comment Properties** | CTRL+ENTER; ALT+ENTER | Opens the Comment Properties dialog box for a comment step. |
| | **Step Properties > Object Properties** | CTRL+ENTER; ALT+ENTER | Opens the Object Properties dialog box for a selected object. |
| | **Step Properties > Checkpoint Properties** | | Opens the relevant Checkpoint Properties dialog box for a selected object. |
| | **Step Properties > Output Value Properties** | | Opens the relevant Output Value Properties dialog box for a selected object. |
| | **Step Properties > Report Properties** | CTRL+ENTER; ALT+ENTER | Displays the Report Properties dialog box for a report step. |
|  | **Find** | CTRL+F | Searches for a specified string. |
|  | **Replace** | CTRL+H | Searches and replaces a specified string. |
|  | **Go To** | CTRL+G | Moves the cursor to a particular line in the component. |
| | **Bookmarks** | CTRL+B | Creates bookmarks in your script for easy navigation. |
|  | **Advanced > Comment Block** | CTRL+M | Comments out the current row, or selected rows. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Advanced > Uncomment Block** | CTRL+SHIFT+M | Removes the comment formatting from the current or selected rows. |
|  | **Advanced > Indent** | TAB | Indents the step according to the tab spacing defined in the Editor Options dialog box. |
|  | **Advanced > Outdent** | BACKSPACE | Outdents the step (reduces the indentation) according to the tab spacing defined in the Editor Options dialog box. |
| | **Advanced > Go to Function Definition** | ALT+G | Navigates to the definition of the selected function. |
| | **Advanced > Complete Word** | CTRL+SPACE | Completes the word when you type the beginning of a VBScript method or object. |
| | **Advanced > Argument Info** | CTRL+SHIFT+ SPACE | Displays the syntax of a method. |
| | **Advanced > Apply "With" to Script** | CTRL+W | Generates With statements for the action displayed in the Expert View. |
| | **Advanced > Remove "With" Statements** | CTRL+SHIFT+W | Converts any With statements in the action displayed in the Expert View to regular (single-line) VBScript statements. |
| | **Optional Step** | | Inserts an optional step (a step that is not required to successfully complete a run session). |

Many of the **Edit** menu commands are also available from the **Edit** toolbar (described on page 69).

## View Menu Commands

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

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

Some of the **View** menu commands are also available from the **View** toolbar (described on page 70).

## Insert Menu Commands

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Checkpoint > Standard Checkpoint** | F12 | Opens the Checkpoint Properties dialog box, enabling you to create a standard checkpoint for an object. |
|  | **Output Value > Standard Output Value** | CTRL+F12 | Opens the Output Value Properties dialog box, enabling you to create a standard output value for an object. |
|  | **Step Generator** | F7 | Opens the Step Generator. (Relevant only for function libraries) |
|  | **Function Definition Generator** | | Opens the Function Definition Generator. (Relevant only for function libraries) |
| | **New Step** | F8; INSERT | Inserts a new step in the Keyword View. |
|  | **Operation** | | Inserts an operation (function) step in a component. |
|  | **Comment** | | Inserts a Comment step in the Keyword View. |

Some of the **Insert** menu commands are also available from the **Insert** toolbar (described on page 70).

## Automation Menu Commands

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Record** | F3 | Starts a recording session. |
|  | **Run** | F5 | Starts a run session from the beginning or from the line at which the session was paused. |
|  | **Stop** | F4 | Stops the recording or run session. |
| | **Run Current Action** | | Runs only the active action. |
|  | **Run from Step** | CTRL+F5 | Starts a run session from the selected step. |
|  | **Maintenance Run Mode** | | Starts a run session during which the Maintenance Run Mode wizard opens for steps that failed because an object was not found in the application (if applicable). |
|  | **Update Run Mode** | | Starts a run session to update test object descriptions and other options (if applicable). |
|  | **Analog Recording** | SHIFT+ALT+F3 | Starts recording in **Analog Recording** mode. (Relevant only for tests) |
|  | **Low Level Recording** | CTRL+SHIFT +F3 | Starts recording in **Low Level Recording** mode. (Relevant only for tests) |

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

Some of the **Automation** menu commands are also available from the **Automation** toolbar (described on page 69).

## Resources Menu Commands

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Object Repository** | CTRL+R | Opens the Object Repository window, which displays a tree containing all objects in the current test or component. |
|  | **Object Repository Manager** | | Opens the Object Repository Manager dialog box, enabling you to open and modify multiple shared object repositories. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Associate Repositories** | | Opens the Associate Repositories dialog box, enabling you to manage the object repository associations for the test. |
|  | **Map Repository Parameters** | | Opens the Map Repository Parameters dialog box, enabling you to map repository parameters, as needed. |
| | **Recovery Scenario Manager** | | Opens the Recovery Scenario Manager dialog box. |
| | **Associated Function Libraries** | | Lists the function libraries associated with the active document, enabling you to open them. |

The **Object Repository** menu command is also available from the **Standard** toolbar (described on page 68).

## Debug Menu Commands

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Pause** | | Pauses the debug session. |
|  | **Step Into** | F11 | Runs only the current line of the script. If the current line calls a method, the method is displayed in the view but is not performed. |
|  | **Step Over** | F10 | Runs only the current line of the script. When the current line calls a method, the method is performed in its entirety, but is not displayed in the view. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Step Out** | SHIFT+F11 | Runs to the end of the method then pauses the run session. (Available only after running a method using **Step Into**.) |
| | **Run to Step** | CTRL+F10 | Runs until the current step. |
| | **Debug from Step** | | Runs from the selected step instead of the start of the component. |
| | **Add to Watch** | CTRL+T | Adds the selected item to the Watch tab. |
|  | **Insert/Remove Breakpoint** | F9 | Sets or clears a breakpoint in the component. |
|  | **Enable/Disable Breakpoint** | CTRL+F9 | Enables or disables a breakpoint in the component. |
|  | **Clear All Breakpoints** | CTRL+SHIFT+ F9 | Deletes all breakpoints in the component. |
|  | **Enable/Disable All Breakpoints** | | Enables or disables all breakpoints in the component. |

Some of the **Debug** commands are also available from the **Debug** toolbar (described on page 69).

## Tools Menu Commands

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Options** | | Opens the Options dialog box, enabling you to modify global testing options. |
| | **View Options** | | Opens the Editor Options dialog box, enabling you to customize how tests and function libraries are displayed in the Expert View and Function Library windows. |
|  | **Check Syntax** | CTRL+7 | Checks the syntax of the active document. |
| | **Object Identification** | | Opens the Object Identification dialog box, enabling you to specify how QuickTest identifies a particular test object. |
|  | **Object Spy** | | Opens the Object Spy dialog box, enabling you to view the run-time or test object properties and methods of any object in an open application. |
| | **Web Event Recording Configuration** | | Opens the Web Event Recording Configuration dialog box, enabling you to specify a recording configuration level. (Relevant for tests only) |
| | **Data Driver** | | Opens the Data Driver dialog box, which displays the default Constants list for the action. (Relevant for tests only) |
| | **Change Active Screen** | | Replaces the previously recorded Active Screen with the selected Active Screen. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
| | **Virtual Objects > New Virtual Object** | | Opens the Virtual Object Wizard, enabling you to teach QuickTest to recognize an area of your application as a standard test object. |
| | **Virtual Objects > Virtual Object Manager** | | Opens the Virtual object Manager, enabling you to manage all of the virtual object collections defined on your computer. |

Some of the **Tools** menu commands are also available from the **Tools** toolbar (described on page 70).

## Window Menu Commands

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

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

## Help Menu Commands

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

| Command | Shortcut Key | Function |
|---|---|---|
| **QuickTest Professional Help** | F1 | Opens the QuickTest Professional Help. |
| **Printer-Friendly Documentation** | | Opens a page that provides links to printer-friendly versions of all QuickTest documentation, in Adobe Acrobat Reader (PDF) format. |
| **QuickTest Professional Tutorial** | | Opens the QuickTest Professional tutorial, which teaches you basic QuickTest skills and shows you how to start testing your applications. |
| **What's New** | | Opens the What's New in QuickTest Professional Help. |
| **Product Feature Movies** | | Enables you to view movies illustrating various QuickTest features. |
| **Knowledge Base** | | Opens the Knowledge Base area of the HP Customer Support Site, enabling you to view product-specific knowledge base articles. (Requires login) |

| Command | Shortcut Key | Function |
|---|---|---|
| **Customer Support Web Site** | | Opens the HP Customer Support Web site. This site enables you to browse the HP Support Knowledge Base and add your own articles. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more.<br>The URL is: www.hp.com/go/hpsoftwaresupport |
| **Send Feedback** | | Opens the HP Customer Support Site, enabling you to send feedback about QuickTest Professional. |
| **Check for Updates** | | Checks online for any available updates to QuickTest Professional. You can choose which updates you want to download and (optionally) install. |
| **HP Software web site** | | Uses your default Web browser to access 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.<br>The URL is: www.hp.com/managementsoftware |
| **About QuickTest Professional** | | Displays information about the installed version of QuickTest Professional. |

## Data Table Menu Commands

You can perform the following **Data Table** menu commands by pressing the corresponding shortcut keys when one or more cells are selected in the Data Table:

| Command | Shortcut Key | Function |
|---|---|---|
| **Edit** > **Cut** | CTRL+X | Cuts the table selection and puts it on the Clipboard. |
| **Edit** > **Copy** | CTRL+C | Copies the table selection and puts it on the Clipboard. |
| **Edit** > **Paste** | CTRL+V | Pastes the contents of the Clipboard to the current table selection. |
| **Edit** > **Clear** > **Contents** | CTRL+DEL | Clears the contents from the current selection. |
| **Edit** > **Insert** | CTRL+I | Inserts empty cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells. |
| **Edit** > **Delete** | CTRL+K | Deletes the current selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells. |
| **Edit** > **Fill Right** | CTRL+R | Copies data in the left-most cell of the selected range to all cells to the right of it, within the selected range. |
| **Edit** > **Fill Down** | CTRL+D | Copies data in the top cell of the selected range to all cells below it within the selected range. |
| **Edit** > **Find** | CTRL+F | Finds a cell containing specified text. You can search the table by row or column and specify to match case or find entire cells only. |

| Command | Shortcut Key | Function |
|---|---|---|
| **Edit** > **Replace** | CTRL+H | Finds a cell containing specified text and replaces it with different text. You can search the table by row or column and specify to match case and/or to find entire cells only. You can also replace all. |
| **Data** > **Recalc** | F9 | Recalculates the selected data in the Data Table. |
| Switch between Data Table sheets | CTRL+PAGE UP/PAGE DOWN | Switches through the Data Table sheets when the Data Table is in focus. |

## Other QuickTest Commands

You can perform the following special options using shortcut keys:

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

| Option | Shortcut Key | Function |
|---|---|---|
| Collapse branch | - *[on the numeric keypad]* | Collapses the selected item branch and all branches below it in the Keyword View. |
| Open the **Item** or **Operation** list | SHIFT+F4 or SPACE, when the **Item** or **Operation** column is selected in the Keyword View. | Opens the **Item** or **Operation** list in the Keyword View, when the **Item** or **Operation** column is selected. |

## Browsing the QuickTest Professional Program Folder

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

---

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

---

- **Tools.** Contains the following utilities and tools that assist you with the testing process:
  - **Action Conversion Tool.** Enables you to convert test actions that were created using QuickTest Professional to scripted components for use in business process testing. For more information, see "Converting an Action to a Scripted Component" on page 482 or click the **Help** button in the Action Conversion Tool window.
  - **Additional Installation Requirements.** Opens the Additional Installation Requirements dialog box, which displays any prerequisite software that you must install or configure to work with QuickTest.

- **Business Component Upgrade Tool.** Opens the Business Component Upgrade Tool. If you are connected to a Quality Center project, this tool enables you to upgrade all of the business components in a Quality Center project, from an earlier component version to the format required by the current version. For more information, click the **Help** button in the Business Component Upgrade Tool window.
- **HP Micro Player.** Opens the HP Micro Player, which enables you to view captured movies of a run session without opening QuickTest. For more information, click the **Help** button in the HP Micro Player window.
- **License Validation Utility.** Opens the License Validation utility, which enables you to retrieve and validate license information. For more information, click the **Help** button in the License Validation Utility window.
- **Password Encoder**. Opens the Password Encoder dialog box, which enables you to encode passwords. You can use the resulting strings as method arguments or Data Table parameter values (tests only). For more information, see "Inserting Encoded Passwords into Method Arguments" on page 528.
- **QuickTest Script Editor.** Opens the QuickTest Script Editor, which enables you to open and modify the scripts of multiple tests and function libraries, simultaneously. For more information, see the *HP QuickTest Professional User's Guide*.
- **Register New Browser Control**. Opens the Register Browser Control Utility, which enables you to register your browser control application so that QuickTest Professional recognizes your Web object when recording or running tests. For more information, see the section on registering browser controls in the *HP QuickTest Professional Add-ins Guide*.
- **Remote Agent.** Activates the QuickTest Remote Agent, which enables you to configure how QuickTest behaves when a component is run by a remote application such as Quality Center. For more information, see the *HP QuickTest Professional User's Guide*.
- **Save and Restore Settings**. Opens the Save and Restore Settings dialog box, which enables you to save your existing configurations before uninstalling an old version, and then restore them after installing a new version. For more information, see the *HP QuickTest Professional User's Guide*.

- **Silent Test Runner.** (Relevant only for tests) Opens the Silent Test Runner dialog box, which enables you to run a QuickTest test the way it is run from LoadRunner and Business Availability Center. For more information, see the *HP QuickTest Professional User's Guide*.
- **Test Batch Runner.** (Relevant only for tests) Opens the Test Batch Runner dialog box, which enables you to set up QuickTest to run several tests in succession. For more information, see the *HP QuickTest Professional User's Guide*.
- **Test Results Deletion Tool.** Opens the Test Results Deletion Tool dialog box, which enables you to delete unwanted or obsolete results from your system according to specific criteria that you define. For more information, see "Deleting Results Using the Test Results Deletion Tool" on page 651.

- **Documentation.** Provides the following links to commonly used documentation files:
  - **Printer-Friendly Documentation.** Opens a page that provides links to printer-friendly versions of all QuickTest documentation, in Adobe Acrobat Reader (PDF) format.
  - **QuickTest Professional Code Samples Plus.** Opens the QuickTest Professional Code Samples Plus Help, which provides sample function libraries, code, and SDK samples with accompanying explanations.
  - **QuickTest Professional Help.** Opens a comprehensive help file containing the *HP QuickTest Professional User's Guide*, the *HP QuickTest Professional for Business Process Testing User's Guide*, *HP QuickTest Professional Add-ins Guide*, the *HP QuickTest Professional Object Model Reference* (including the relevant sections for any installed add-ins), *QuickTest Advanced References* (Automation API and XML Schema references), and the *Microsoft VBScript Reference*.
  - **Tutorial.** Opens the QuickTest Professional tutorial, which teaches you basic QuickTest skills and shows you how to start testing your applications.

- **QuickTest Automation Reference**. Opens the QuickTest Automation Reference. The automation object model assists you in automating test management, by providing objects, methods and properties that enable you to control QuickTest features and configurations. The QuickTest Automation Reference provides syntax, descriptive information, and examples for the objects, methods, and properties. It also contains a detailed overview to help you get started writing QuickTest automation scripts.

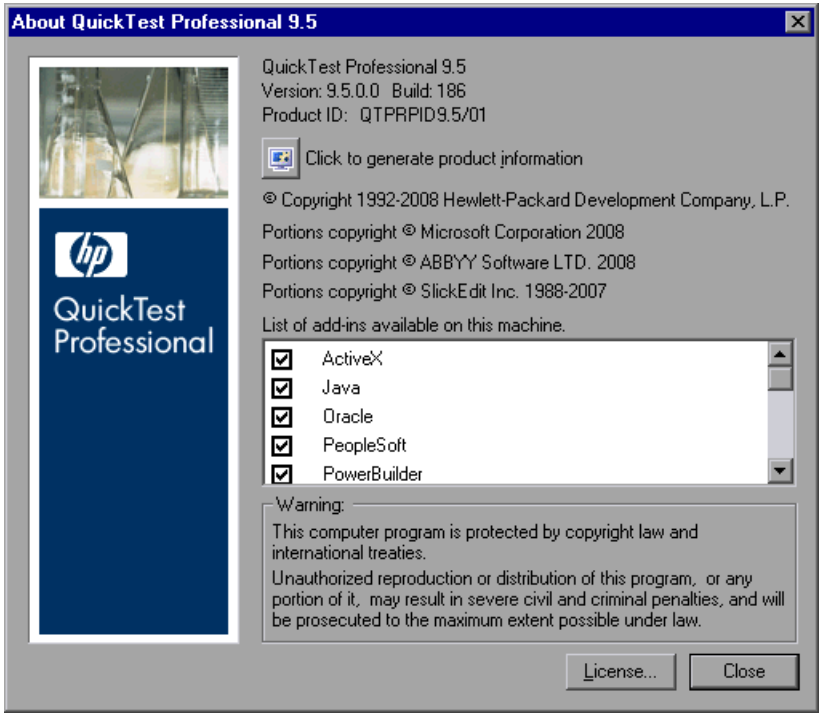
- **Sample Applications**. Contains the following links to sample applications that you can use to practice testing with QuickTest:
  - **Flight**. Opens a sample flight reservation Windows application. To access the application, enter any username and the password **mercury**.
  - **Mercury Tours Web Site**. Opens a sample flight reservation Web application. This Web application is used as a basis for the QuickTest tutorial. For more information, see the *HP QuickTest Professional Tutorial*.
- **QuickTest Professional**. Opens the QuickTest Professional application.
- **Readme**. Opens the *HP QuickTest Professional Readme*, which provides the latest news and information on QuickTest Professional and the QuickTest Professional add-ins.
- **Test Results Viewer**. Opens the Test Results window, which enables you to select a component or business process test and view information about the steps performed during the run session. For more information, see "The Test Results Window" on page 627.
- **Check for Updates**. Checks online for any available updates to QuickTest Professional. You can choose which updates you want to download and (optionally) install.

# Viewing Product Information

You can view information regarding the QuickTest add-ins and hotfixes (patches) installed on your computer, as well as about your operating system. This information is useful for troubleshooting and when dealing with HP Customer Support.

**To view the product information:**

**1** In QuickTest, choose **Help** > **About QuickTest Professional**. The About QuickTest Professional 9.5 window opens.



The About QuickTest Professional 9.5 window displays the following information:

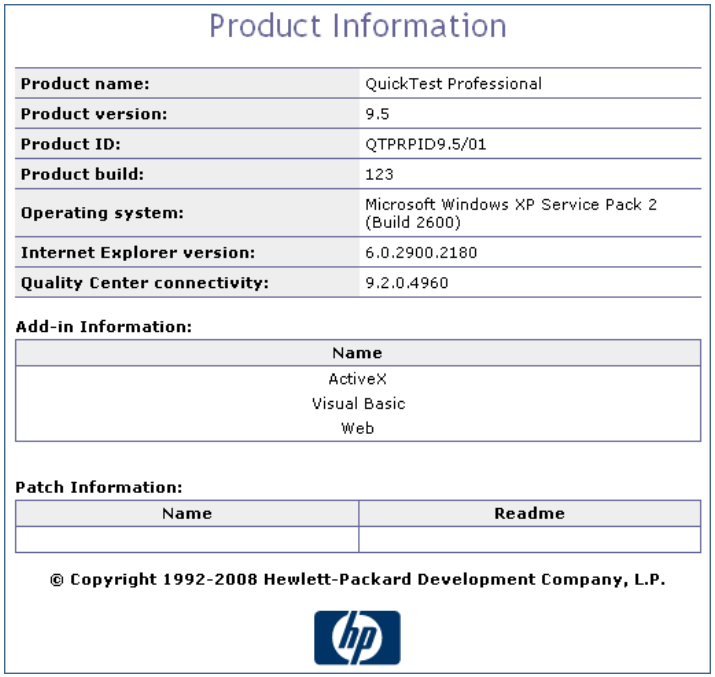
- The version of QuickTest that is installed on your computer, its build number, and Product ID number.
- The list of QuickTest add-ins that are installed on your computer. A check mark next to the add-in name indicates that the add-in is currently loaded. For more information on QuickTest add-ins, see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

---

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

---

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



Product Information

| | |
|---|---|
| **Product name:** | QuickTest Professional |
| **Product version:** | 9.5 |
| **Product ID:** | QTPRPID9.5/01 |
| **Product build:** | 123 |
| **Operating system:** | Microsoft Windows XP Service Pack 2 (Build 2600) |
| **Internet Explorer version:** | 6.0.2900.2180 |
| **Quality Center connectivity:** | 9.2.0.4960 |

**Add-in Information:**

| Name |
|---|
| ActiveX |
| Visual Basic |
| Web |

**Patch Information:**

| Name | Readme |
|---|---|
| | |



The Product Information window displays the following information:

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

# Part II

## Working with Test Objects and Object Repositories

# 3

# Understanding the Test Object Model

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

**This chapter includes:**



## About Understanding the Test Object Model

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

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

## Understanding How QuickTest Learns Objects

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

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

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

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

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

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

QuickTest uses a very similar method when it learns objects.

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

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

## Understanding How QuickTest Identifies Objects During the Run Session

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

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

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

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

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

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

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

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

QuickTest uses a very similar process of elimination with its **Smart Identification** mechanism to identify an object, even when the learned description is no longer accurate. Even if the values of your test object properties change, QuickTest maintains your component's reusability by identifying the object using Smart Identification. For more information on Smart Identification, see Chapter 5, "Configuring Object Identification."

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

## Applying the Test Object Model Concept

The test object model is a large set of object types or classes that QuickTest uses to represent the objects in your application. Each test object class has a list of properties that can uniquely identify objects of that class and a set of relevant methods that QuickTest can learn about it.

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

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

QuickTest learns objects when it adds them to an **object repository**, which is a storehouse for objects. You can add objects to an object repository in several ways. For example, you can use the QuickTest Navigate and Learn option, add objects manually, or perform an operation on your application while recording. For more information on object repositories, see Chapter 6, "Managing Object Repositories" and Chapter 10, "Creating Tests Using the Keyword-Driven Methodology."

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

- identifies the QuickTest test object class that represents the learned object and creates the appropriate test object
- reads the current value of the object's properties in your application and stores the list of properties and values with the test object
- chooses a unique name for the object, generally using the value of one of its prominent properties

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

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

QuickTest identifies the object as a WebButton test object. In the object repository, QuickTest creates a WebButton object with the name Search, and learns the following properties and values for the **Search** WebButton:

| Name | Value |
|---|---|
| Description properties | |
| type | submit |
| name | Search |
| html tag | INPUT |

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

Suppose that clicking the Search button is the first step in your component, QuickTest displays your step as follows:

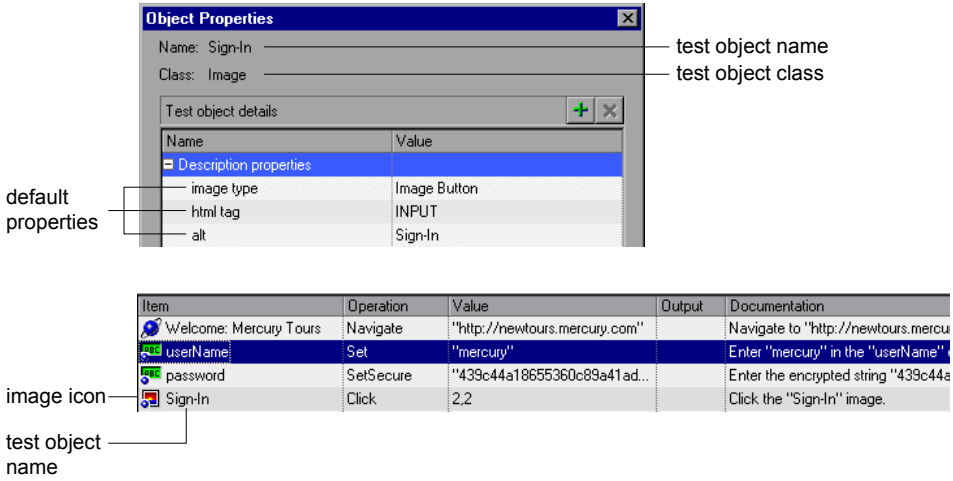
| Item | Operation | Documentation |
|---|---|---|
| Search | Click | Click the "Search" button. |

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

## Understanding Test Object Descriptions

For each object class, QuickTest learns a set of properties when it learns an object. QuickTest then uses this description to identify the object when it runs the component.

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




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

When the component runs, QuickTest searches for the object that matches the description it learned. If it cannot find any object that matches the description, or if it finds more than one object that matches, QuickTest may use the Smart Identification mechanism to identify the object.

You can configure the mandatory, assistive, and ordinal identifier properties that QuickTest uses to learn the descriptions of the objects in your application, and you can enable and configure the Smart Identification mechanism. For more information, see Chapter 5, "Configuring Object Identification."

## Understanding Test Object and Run-Time Object Properties and Methods

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

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

During a run session, QuickTest performs the specified test object method on the run-time object. Run-time object methods are the methods of the object in your application as defined by the object creator. You can access and perform run-time object methods using the Object property.

For information on activating run-time methods using the Object property, see "Retrieving and Setting Test Object Property Values" on page 352.

Test object properties are the properties whose values are captured from the objects in your application when QuickTest learns the object. QuickTest uses the values of these properties to identify run-time objects in your application during a run session.

Property values of objects in your application may change dynamically each time your application opens, or based on certain conditions. You may need to modify the test object property values to match the run-time object property values. You can modify test object properties manually while designing your component, or use SetTOProperty statements during a run session (via an operation defined in a function library). You can also use regular expressions to identify property values based on conditions or patterns you define. For more information on modifying object properties, see Chapter 4, "Working with Objects."

You can view or modify the test object property values that are stored with your component in the Object Properties or Object Repository dialog box. For more information, see "Modifying Test Object Properties" on page 134.

You can view the current test object property values of any object on your desktop using the Properties tab of the Object Spy. For more information, see "Viewing Object Properties and Methods Using the Object Spy" on page 108.

You can view the syntax of the test object methods as well as the run-time methods of any object on your desktop using the Methods tab of the Object Spy. For more information, see "Viewing Object Properties and Methods Using the Object Spy" on page 108.

Using operations defined in function libraries, you can retrieve or modify property values of the test object during the run session by adding GetTOProperty and SetTOProperty statements. You can retrieve property values of the run-time object during the run session by adding GetROProperty statements. For more information, see "Retrieving and Setting Test Object Property Values" on page 352.

If the available test object methods or properties for an object do not provide the functionality you need, you can access the internal methods and properties of any run-time object using the Object property. You can also use the attribute object property to identify Web objects in your application according to user-defined properties. For information, see "Accessing Run-Time Object Properties and Methods" on page 354.

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

# Viewing Object Properties and Methods Using the Object Spy

Using the Object Spy pointing hand mechanism, you can view the supported properties and methods of any object in an open application. As you move the pointing hand over the objects in the application, their details are displayed in the Object Spy. These details may include the test object's hierarchy tree, its properties and values, and the methods associated with the object. For methods, the syntax is also displayed. In most environments, you can choose to view the test object properties and methods or the run-time (native) properties and methods.

In some environments, the test object is also highlighted in the application as you move the pointing hand over it. This enables you to visually distinguish between the various test objects in the application.

**To view test object or run-time object properties or methods:**

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

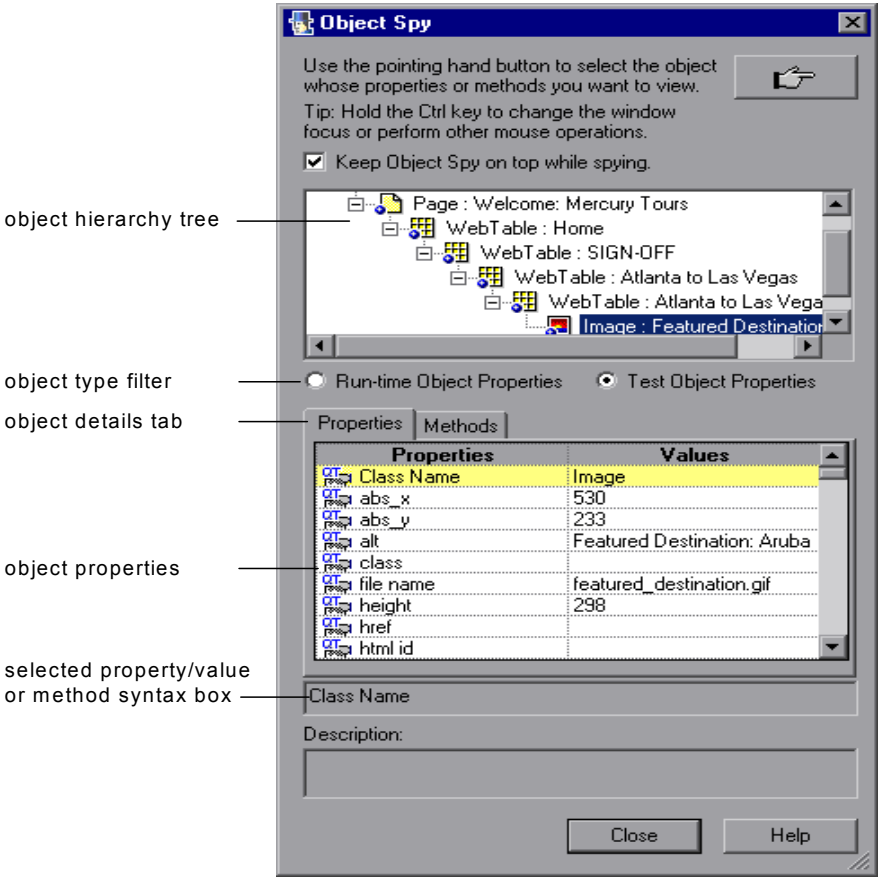
**2** Choose **Tools** > **Object Spy** or click the **Object Spy** toolbar button to open the Object Spy dialog box and display the **Properties** tab. Alternatively, click the **Object Spy** button from the Object Repository dialog box. For more information on the Object Repository dialog box, see "Understanding the Object Repository Window" on page 120.

**3** Select the details you want to view for the object.

- Click **Run-time Object Properties** or **Test Object Properties**.
- To view the object's available methods and syntax, click the **Methods** tab. Otherwise, the **Properties** tab is displayed by default, enabling you to view the object's properties and their values.

**4** In the Object Spy dialog box, click the pointing hand. QuickTest is hidden. As you move the pointing hand over the test objects in your application, the test objects are highlighted, and you can view their test object properties or methods in the Object Spy dialog box. You can also view their parent objects in the object hierarchy tree area of the Object Spy dialog box. For tips on changing the window focus, viewing partially or fully hidden windows, and working with the Object Spy, see "Tips for Working with the Object Spy" on page 111.

**5** Highlight or click the object whose properties or methods you want to view. The Object Spy displays the object hierarchy tree and the properties or methods of the object that is selected within the tree.




---

**Note:** The example above shows the object hierarchy tree after clicking an object. While an object is highlighted, objects names, such as Atlanta to Las Vegas and Featured Destinations, are not displayed.

---

**6** Click the object whose associated methods you want to view. The Object Spy displays the object hierarchy tree and details for the selected object according to your selection. which can be the run-time object or test object properties/values or methods associated with the object that is selected in the tree.

**7** To view the properties or methods of the test object, click the **Test Object Properties** radio button. To view the properties or methods of the run-time object, click the **Run-time Object Properties** radio button.

---

**Tips:**

In a function, you can use the Object property to retrieve the values of the run-time properties displayed in the Object Spy or to activate the run-time object methods. For more information, see "Retrieving Run-Time Object Properties" on page 354 and "Activating Run-Time Object Methods" on page 355.

In a function, you can also use the GetTOProperty and SetTOProperty methods to retrieve and set the value of test object properties for test objects. You can use the GetROProperty to retrieve the current value of the properties of objects in your application during the run session. For more information, see "Retrieving and Setting Test Object Property Values" on page 352.

---

**8** If you want to view properties, values, or methods for another object within the displayed tree, highlight or click the object in the tree and select the relevant options, as described in step3 on page 108.

**9** If you want to copy an object property or value, or a method's syntax to the Clipboard, click the property, value, or method to highlight it. The value is displayed in the **selected property/value or method syntax** box (located above the **Description** box). Highlight the text in the box and use CTRL + C to copy the text to the Clipboard or right-click the highlighted text and choose **Copy** from the menu.

## Tips for Working with the Object Spy

- If the window on which you want to spy is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds. The window comes into the foreground. You can now point and click on the required object. You can configure the length of time required to bring a window into the foreground using the General tab of the Options dialog box. For more information, see Chapter 20, "Setting Global Testing Options."
- You can hold the left CTRL key to change the window focus.
- If the window on which you want to spy is fully hidden by another window, or if the Object Spy 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 on which you want to spy 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 on which you want to spy is displayed, release the left CTRL key. The pointer becomes a pointing hand again.
- You can view the properties and methods of the test objects in the object hierarchy tree in the Object Spy dialog box by holding the left CTRL key, and clicking the relevant test object.
- You can scroll through the properties and methods of a test object in the Object Spy dialog box by holding the left CTRL key, and using the scroll bar.
- You can resize the Object Spy dialog box. This is useful if you have a deep hierarchy, or long property names and values, enabling you view all the information without scrolling.

# 4

# Working with Objects

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

**This chapter includes:**

# About Working with Objects

When QuickTest runs a component, it simulates a human user by moving the pointer over the application, clicking objects, and entering keyboard input. Like a human user, QuickTest must learn the interface of an application to be able to work with it. QuickTest does this by learning the application's objects and their corresponding property values and storing these object descriptions in an **object repository**.

QuickTest also stores **checkpoint objects** and **output objects** in the object repository.

As QuickTest learns the test objects, it stores them in the component's **local object repository**. You can choose to keep the stored test objects in the local object repository, or you can choose to store the test objects in a **shared object repository**. Storing the test objects in the local object repository makes them available only to the specific component, but not to other components. Storing the test objects in one or more shared object repositories enables multiple components (via their application areas) to use them.

You can export your local object repository to a shared object repository. For more information on exporting and replacing local objects, see "Exporting Local Objects to a Shared Object Repository" on page 178.

You can also work with a combination of local and shared object repositories, as needed. For more information on local and shared object repositories, see "Understanding Object Repository Types" on page 115.

If one or more of the property values of a test object in your application differ from the property values QuickTest uses to identify the test object, your component may fail. Therefore, when the property values of objects in your application change, you should modify the corresponding test object property values in the corresponding object repository so that you can continue to use your existing components. For more information on maintaining and updating your components, see "Maintaining Components" on page 701.

You can modify objects stored in a local object repository using the Object Repository window, as described in this chapter. You can modify objects in a shared object repository using the Object Repository Manager. For information on the Object Repository Manager, see Chapter 6, "Managing Object Repositories." You can also copy objects from a shared object repository to a local object repository and then modify the local copy of the object using the Object Repository window, as described in this chapter.

You can also manage some aspects of a local object repository using the QuickTest Object Repository automation object model. For example, you can add, remove, and rename objects in the local object repository. For more information, see "Managing Object Repositories Using Automation" on page 245.

## Understanding Object Repository Types

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

When you plan and create components, you must consider how you want to store the objects in your components. You can store the objects for each component in its corresponding local object repository, or you can store the objects in your components in one or more shared object repositories. By storing objects in shared object repositories and associating these repositories with your components' application areas, you enable multiple components to use the objects. For each component, you can use a combination of objects from your local and shared object repositories, according to your needs. You can also transfer local objects to a shared object repository, if required. This reduces maintenance and enhances the reusability of your components because it enables you to maintain the objects in a single, shared location instead of multiple locations. For more information, see "Deciding Whether to Use Local or Shared Object Repositories" on page 117.

If you are new to using QuickTest, you may want to use local object repositories. In this way, you can record and run 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 component. If you modify an object in the local object repository, your changes do not have any effect on any other component.

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 components—if the components include the same objects. Object information that applies to many components is kept in one central location. When the objects in your application change, you can update them in one location for all the components that use this shared object repository.

If an object with the same name is located in both the local object repository and in a shared object repository associated with the same component, the component uses the local object definition. If an object with the same name is located in more than one shared object repository associated with the same 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 component. For more information on associating shared object repositories, see "Managing Shared Object Repositories" on page 432.

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

When you open and work with an existing component, it always uses the object repositories that are specified in the application area with which the component is associated. Shared object repositories are read-only when accessed from components; you edit them using the Object Repository Manager.

## Deciding Whether to Use Local or Shared Object Repositories

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

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

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

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

- You are creating components using keyword-driven methodologies (not by recording).
- You have several components that test elements of the same application, interface, or set of objects.
- You expect the object properties in your application to change from time to time and/or you regularly need to update or modify object properties.

### Understanding the Local Object Repository

When you use a local object repository, QuickTest uses a separate object repository for each component. (You can also use one or more shared object repositories if needed. For more information, see "Understanding the Shared Object Repository" on page 119.) The local object repository is fully editable from within its component.

When working with a local object repository:

- QuickTest creates a new (empty) object repository for each component.

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

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

- If a child object is added to a local object repository, and its parents are in a shared object repository, its parents are automatically moved to the local object repository.

- Every time you create a new component, QuickTest creates a new, corresponding local object repository and adds test objects to the repository as it learn them.

- If QuickTest learns the same object in your application in two different components, the test object is stored as a separate test object in each of the local object repositories.

- When you save your component, the local object repository is automatically saved with the it. The local object repository is not accessible as a separate file (unlike the shared object repository).

## Understanding the Shared Object Repository

When you use shared object repositories, QuickTest uses the shared object repositories you specified for the selected 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 component if you need to access them only from the specific component. For more information, see "Understanding the Local Object Repository" on page 117.)

After you begin creating your component, you can specify additional shared object repositories. You can also create new ones and associate them with your component. Before running the component, you must ensure that the object repositories being used by the component contain all of the objects in your component. Otherwise, the component may fail. For more information, see "Adding Test Objects to an Object Repository" on page 156.

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

When working with a shared object repository:

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

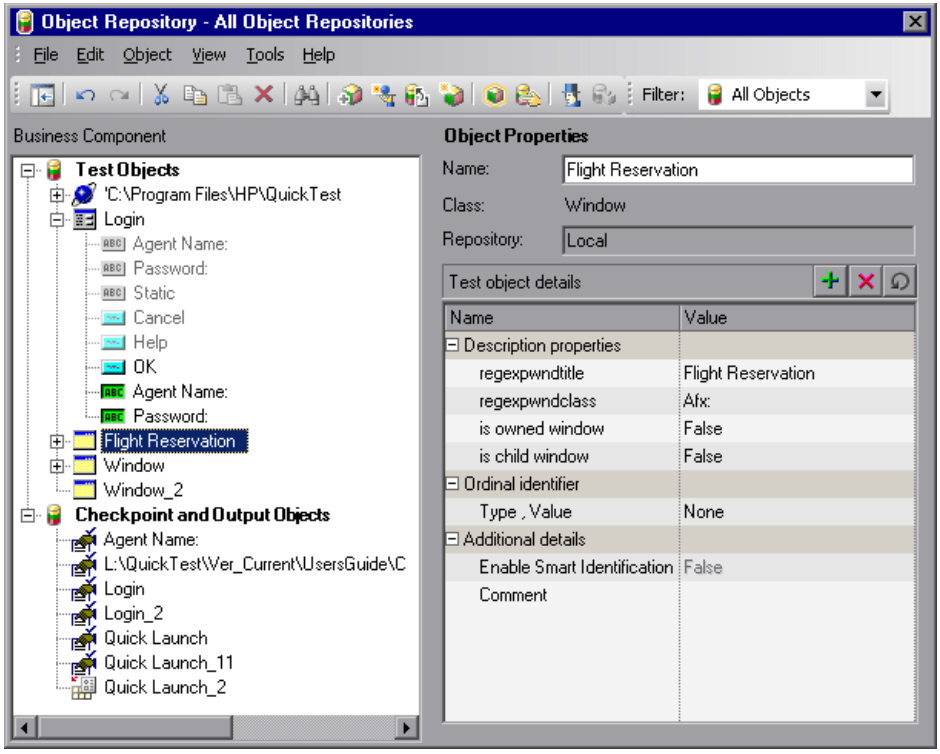
You can export objects from the local object repository to a shared object repository. This enables you to make the local objects accessible to other components. For more information, see "Exporting Local Objects to a Shared Object Repository" on page 178.

You can also merge objects from the local object repository directly to a shared object repository that is associated with the same component. This can help reduce maintenance since you can maintain the objects in a single shared location, instead of multiple locations. For more information, see "Updating a Shared Object Repository from Local Object Repositories" on page 268.

## Understanding the Object Repository Window

You open the Object Repository window for a specific component by choosing **Resources** > **Object Repository** or clicking the **Object Repository** button.

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




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

---

**Note:** Test objects of environments that are not installed with QuickTest will be displayed with an unknown icon (question mark) in the object repository.

---

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

---

**Note:** You can choose whether to show only the object repository tree, or the object repository tree together with the object details area. For more information, see "Showing and Hiding the Object Details Area" on page 129.

---

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

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

You can also drag and drop test objects from other locations. For more information, see:

- "Understanding the Available Keywords Pane" on page 755.
- "Adding Test Objects to Your Component Using the Object Repository Manager" on page 227.

For more information on viewing and modifying object properties, see "Modifying Test Object Properties" on page 134.

---

**Notes:**

- 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** options or **Undo** and **Redo** buttons to cancel or repeat your changes. When you save the current component, you cannot undo or redo operations that were performed before the save operation.
- Even when steps containing an object are deleted from your component, the objects remain in the object repository. You can delete objects from the local object repository using the Object Repository window. You can delete objects from a shared object repository using the Object Repository Manager. For more information, see "Managing Object Repositories" on page 209.

---

The Object Repository window contains the following information:

| Information | Description |
|---|---|
| **Business Component** | Indicates that the current testing document is a business component. |
| **Test Objects** tree | Contains all test objects in the current component (all local test objects and all test objects in any shared object repositories associated with the selected component).<br>**Note:** If there are test objects in different associated object repositories with the same name, object class, and parent hierarchy, the object repository tree shows only the first one it finds based on the priority order defined. For information on object repository priorities, see "Managing Shared Object Repositories" on page 432.<br>You can filter the objects shown in the object repository tree. For more information, see "Filtering the Object Repository Window" on page 129. |
| **Checkpoint and Output Objects** tree | Contains all the checkpoint and output objects in the current component (all local checkpoint and output objects and all checkpoint and output objects in any shared object repositories associated with the selected component). |
| **Name** | The name that QuickTest assigns to the object. You can change the name of a object in the local object repository. For more information, see "Renaming Test Objects" on page 140. |
| **Class** | The class of the object. |

| Information | Description |
|---|---|
| **Repository** | The location (file name and path) of the object repository in which the object is located. If the object is located in the local object repository, **Local** is displayed. |
| Object details | Enables you to view the properties and property values used to identify a test object during a run session or the properties of a checkpoint or output object. You can also modify the object details for an object in the local object repository. For more information, see "Understanding the Object Details Area" on page 125. You can choose whether to show or hide the object details area. For more information, see "Showing and Hiding the Object Details Area" on page 129. |

## Understanding the Object Details Area

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

---

**Tip:** You can choose whether to show or hide the object details area. For more information, see "Showing and Hiding the Object Details Area" on page 129.

---

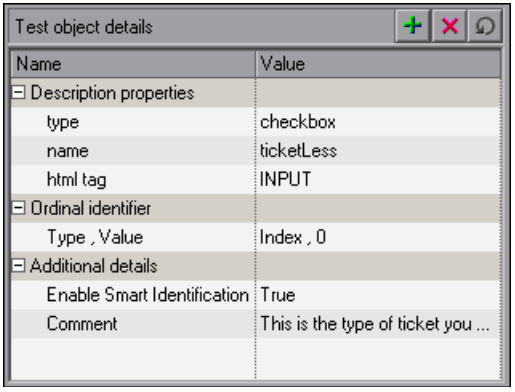
In the Object Repository window, objects in a shared object repository are displayed in the Object Properties pane (including the object details area) in read-only format. To modify objects in a shared object repository, open the shared object repository using the Object Repository Manager. For more information, see Chapter 6, "Managing Object Repositories." You can also modify an object in a shared object repository by copying to the local object repository and then modifying the local copy. For more information, see "Copying an Object to the Local Object Repository" on page 131.

**Tips:**

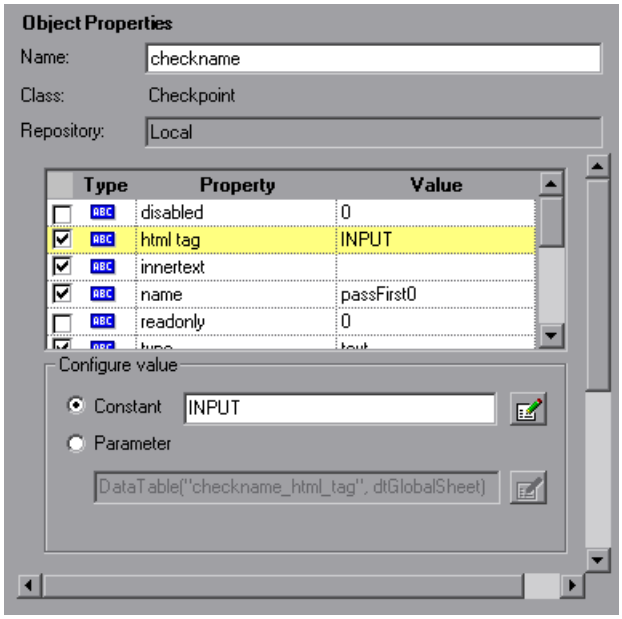
- You can view object properties and property values using the Object Properties dialog box. For more information, see "Viewing Object Properties and Property Values" on page 133.

- You can use the Object Spy at any time to view run-time or test object properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Methods Using the Object Spy" on page 108.

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



You can also modify checkpoint and output value details for objects saved in the local object repository.




You can also copy an object from a shared object repository to the local object repository, and then modify it.

---

**Note:** 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** options or **Undo** and **Redo** buttons to cancel or repeat your changes. When you save the current component, you cannot undo and redo operations that were performed before the save operation.

---

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

| Item | Description |
|---|---|
| **Description properties** | The properties and property values used to identify the object during a run session.<br>You can add and remove properties to or from the test object description. For more information, see "Adding Properties to a Test Object Description" on page 143.<br>You can specify a property value as a constant, or you can parameterize the value. For more information, see "Specifying or Modifying Property Values" on page 136. |
| **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 more information, see "Specifying Ordinal Identifiers" on page 150. |
| **Additional details** | Contains the following options:<br>➤ **Enable Smart Identification.** Enables you to select **True** or **False** to specify whether QuickTest should use Smart Identification to identify the test object during the run session if it is not able to identify the object using the test object description.<br>**Note:** This option is available only if Smart Identification properties are defined for the test object's class in the Object Identification dialog box. For more information on Smart Identification, see "Configuring Smart Identification" on page 196.<br>➤ **Comment.** Enables you to add textual information about the test object. |

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

For more information, see:

- "Checking a Bitmap" on page 566
- "About Outputting Values" on page 571

## Showing and Hiding the Object Details Area

You can choose to work with the Object Repository window in Compact View mode or Full View mode. Compact View mode displays only the object repository tree, while Full View mode displays the object repository tree together with the object details area.

**To change the Object Repository window view mode:**

Perform one of the following, depending on the mode you want to show:

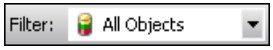
- Choose **View** > **Compact View** or click the **Compact View** button.

- Choose **View** > **Full View** or click the **Full View** button.

The Object Repository window switches to the selected view mode.

## Filtering the Object Repository Window

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
- Only the local objects in the current component
- Only the objects in a specific shared object repository associated with the current 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 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.

## Viewing and Modifying Test Object Properties

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

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

For more information on different ways in which you can view and modify test object properties, see:


- "Copying an Object to the Local Object Repository" on page 131
- "Viewing Object Properties and Property Values" on page 133
- "Modifying Test Object Properties" on page 134
- "Specifying or Modifying Property Values" on page 136
- "Updating Test Object Properties from an Object in Your Application" on page 138
- "Restoring Default Properties for a Test Object" on page 140
- "Renaming Test Objects" on page 140
- "Adding Properties to a Test Object Description" on page 143
- "Defining New Test Object Properties" on page 147
- "Removing Properties from a Test Object Description" on page 149
- "Specifying Ordinal Identifiers" on page 150


## Copying an Object to the Local Object Repository

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

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

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

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

**To copy an object to the local object repository:**

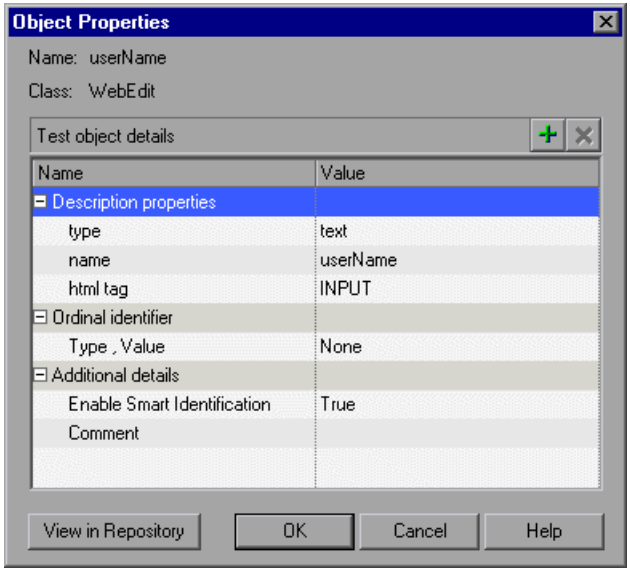
1. In the Object Repository window, select an object from a shared object repository that you want to copy to the local object repository. Objects in a shared object repository are colored gray. You can select more than one object to copy, as long as the selected objects have the same parent objects.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

## Viewing Object Properties and Property Values

You can view test object properties and property values for objects in your component steps.

**To view object properties and property values:**

In your component, click the step of the object whose properties you want to view and choose **Edit** > **Step Properties** > **Object Properties**. The Object Properties dialog box opens.




---

**Note:** There are slight differences in the Object Properties dialog box, depending on whether the selected object is currently stored in the local object repository, the shared object repository, or not stored in any object repository associated with the current component. This section describes options shown in the dialog box for options in the local object repository or not in any associated object repository. For objects stored in a shared object repository, this dialog box appears as for local objects (as shown above), but is in read-only format.

---

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

- View the object's properties and property values—its description properties, ordinal identifier, and other settings.
- Modify the properties and property values used to identify the object (for objects that are stored in the local object repository). You modify the properties and values in the Object Properties dialog box in the same way as you modify the test object details in the Object Repository window. For more information, see "Modifying Test Object Properties" on page 134.
- Click the **View in Repository** button (for objects that are stored in the object repository) to open the Object Repository window and display the selected object in the object hierarchy.
- Click the **Add to Repository** button (for objects that are not stored in the object repository) to add the selected object to the local object repository.

## Modifying Test Object Properties

You can modify a test object by modifying one or more of the test object's property values or by changing the set of properties used to identify that object. You can do this for test objects in the local object repository using the Object Repository window, and for test objects in the shared object repository using the Object Repository Manager.

You can also automatically update the description of one or more test objects in your object repository based on the actual updated object properties in your application. For more information, see "Updating Test Object Properties from an Object in Your Application" on page 138.

---

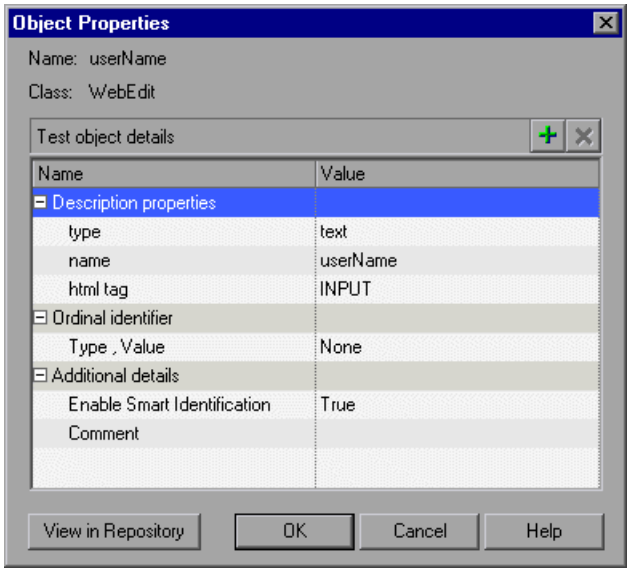
**Tip:** You can use the Object Spy at any time to view run-time or test object properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Methods Using the Object Spy" on page 108.

---

**To modify a test object property:**

**1** Right-click the step containing the test object that changed, and choose **Object Properties** or choose **Edit** > **Step Properties** > **Object Properties** from the menu bar.

The Object Properties dialog box opens and displays the properties QuickTest uses to identify the object.




---

**Tips:**

If you want to view all objects in the 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 component by choosing **Resources** > **Object Repository** or by clicking the **Object Repository** toolbar button.

---

**2** Modify the properties and values as required. You modify the properties and values in the Object Properties dialog box in the same way as you modify the test object details in the Object Repository window. For more information, see "Understanding the Object Details Area" on page 125 and "Viewing and Modifying Test Object Properties" on page 130.

**3** Click **OK** to close the dialog box.

## 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 do this for test objects in the local object repository using the Object Repository window or Object Properties dialog box, and for test objects in the shared object repository using the Object Repository Manager.

You can also find and replace specific test object property values. For more information, see "Finding Objects in an Object Repository" on page 170.

---

**Note:** In some cases, the Smart Identification mechanism may enable QuickTest to identify a test object, even when some of its property values change. However, if you know about property value changes for a specific test object, you should try to correct the test object definition so that QuickTest can identify the test object from its basic object description. For more information on the Smart Identification mechanism, see Chapter 5, "Configuring Object Identification."

---

**Tip:** You can use the Object Spy at any time to view run-time or test object properties and values of the test objects in the application you are testing. You open the Object Spy by choosing **Tools** > **Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Methods Using the Object Spy" on page 108.

---

**To specify a property value:**

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.

---

**Tip:** For a test object in the local object repository, you can also select the required test object and choose **Edit** > **Step Properties** > **Object Properties**, and then make the following property value changes in the Object Properties dialog box.

---

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

- If you want to specify a constant value, enter it in the value cell.
- 
If you want to parameterize the value or specify a constant value using a regular expression, click the parameterization button in the value cell. If you specify a constant value using a regular expression, the  icon is displayed next to the value. For information on parameterizing values, see "Working with Parameters" on page 533.

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

| Parameter Icon | Description |
|---|---|
|  | Indicates that the value of the property is currently a component parameter. |
|  | Indicates that the value of the property is currently a local parameter. |
|  | Indicates that the value of the property is currently a repository parameter (in a shared object repository). |

## Updating Test Object 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, and ordinal identifier and Smart Identification information. Any object-specific comments that you may have entered are not removed.

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

You can also use this option to update an object that you defined (using the **Object** > **Define New Test Object** option) before the application was completely developed, and as a result some of the test object properties and values are missing in the test object description, or are no longer sufficient to identify the object. For more information on the **Define New Test Object** option, see "Defining New Test Objects" on page 164.

You can do this for test objects in the local object repository using the Object Repository window, and for test objects in the shared object repository using the Object Repository Manager.

**To update test object properties from an object in your application:**

1 In the object repository tree, select the test object whose description you want to update.

2 Choose **Object** > **Update from Application** or click the **Update from Application** button. QuickTest is hidden, and the pointer changes into a pointing hand.

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.

**Notes:**

- If the object you want to select is in a window that is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds. The window comes into the foreground. You can now point and click on the object you want. You can configure the length of time required to bring a window into the foreground in the General tab of the Options dialog box. For more information, see Chapter 20, "Setting Global Testing Options." You can also hold the left CTRL key to change the window focus. Additionally, 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 only be displayed 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 changes into a standard pointer and you can perform the event. When the object on which you want to spy is displayed, release the left CTRL key. The pointer becomes a pointing hand again.
- If the location you click is associated with more than one object, the Select an Object dialog box opens. Select an object from the object tree and click **OK**.

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

## Restoring Default 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 in the Object Identification dialog box. Any changes that you have made to the description property set for the test object will be overwritten. However, if property values were defined for any of the mandatory properties they are not modified. 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.

**To restore the mandatory property set:**

**1** In the object repository tree, select the test object whose description you want to restore.

**2** In the **Test object details** area, click the **Restore mandatory property set** button.

**3** Click **Yes** to confirm the operation. The test object's description properties are restored to the mandatory property set for the selected object class.

## Renaming Test Objects

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

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

You rename test objects in the local object repository using the Object Repository window. You rename test objects in the shared object repository using the Object Repository Manager.

If you are working with a shared object repository, your change applies to all occurrences of the test object in all 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 component. If other components in your business process test also include operations on the local test object, you should modify the test object's name in each relevant 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. When you modify the name of a test object in a shared repository, the name is automatically updated in all 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 components that were open at the time. Changes that are saved are also automatically updated in components that use the object repository as soon as you open them. To load and view saved changes in a 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 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 tab of the Options dialog box (**Tools > Options**). 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 component run will fail.

---

---

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

---

**To rename a test object:**

In the object repository tree, select the test object that you want to rename and perform one of the following:

- Choose **Edit** > **Rename** and enter the new name for the test object in the selected node in the tree. Then press ENTER or click anywhere else to remove the focus from the test object.
- Press F2 and enter the new name for the test object.
- In the **Name** box in the Object Properties pane, enter the new name for the test object. Then click anywhere else to remove the focus from the object.

---

**Note:** The name you assign to the test object must be unique within the same class and hierarchy in the object repository. Object names are not case-sensitive.

---

## Adding Properties to a Test Object Description

You can add to the list of properties that QuickTest uses to identify an object. For each object class, QuickTest has a default property set that it uses for the object description for a particular test object. You can use the Add Properties dialog box to change the properties that are included in the test object description. You can do this for test objects in the local object repository using the Object Repository window or Object Properties dialog box, and for test objects in the shared object repository using the Object Repository Manager.

---

**Note:** You can also add any valid test object property to a test object description, even if it does not appear in the Add Properties dialog box. For more information, see "Defining New Test Object Properties" on page 147.

---

Adding to the list of properties is useful when you want to create and run 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 QuickTest did not learn automatically when it learned the object. For example, suppose you are testing a Web site that contains an archive of newsletters. The archive page includes a hypertext link to the current newsletter and additional hypertext links to all past newsletters. The text in the first hypertext link on the page changes as the current newsletter changes, but it always links to a page called **current.html**. Suppose you want to create a step in your 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 QuickTest identifies this hypertext link so that it can continue to find it.

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

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

---

**Tip:** You can use the Object Spy at any time to view run-time or test object properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools** > **Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Methods Using the Object Spy" on page 108.

---

---

**Note:** You can also modify the set of properties that QuickTest learns when it learns objects from a particular object class using the Object Identification dialog box. Such a change generally affects only those objects that QuickTest learns after you make the change. For more information, see "Configuring Object Identification" on page 181. You can also apply the changes you make in the Object Identification dialog box to the descriptions of all objects in an existing component using the **Update Run Mode** option. For more information, see "Updating a Component Using the Update Run Mode Option" on page 720.

---

**To add properties to a test object description:**

**1** In the object repository tree, select the test object whose description you want to modify.

**2** In the **Test object details** area, click the **Add description properties** button.

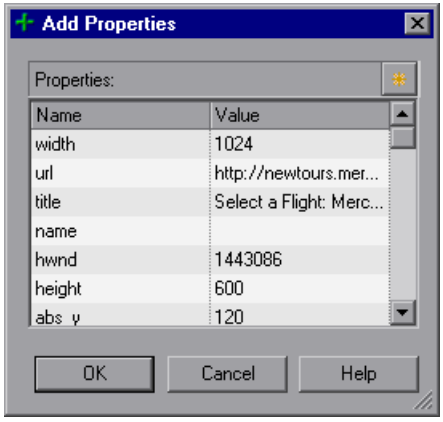
---

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

---

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

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



**Notes:**

- Values for all properties are displayed only if the application that contains the object is currently open. If the application is closed, only values for properties that were part of the object description when the object was learned are shown.
- You can resize the Add Properties dialog box to enable you to view long property values.
-  You can click the **Define new property** button to add valid test object properties to this properties list. For more information, see "Defining New Test Object Properties" on page 147.

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

**Tip:** After you add a new property to the object description, you can modify its value. For more information on modifying object property values, see "Specifying or Modifying Property Values" on page 136.

## Defining New Test Object Properties

You can add any valid test object property to a test object description, even if it does not appear in the Add Properties dialog box. You can do this for test objects in the local object repository using the Object Repository window or Add Properties dialog box, and for test objects in the shared object repository using the Object Repository Manager. For example, suppose you want QuickTest to use a specific property to identify your object, but that property is not listed in the Add Properties dialog box. You can open the Add Properties dialog box and add that property to the list.

---

**Tip:** You can use the Properties tab of the Object Spy to view a complete list of valid test object properties for a selected object. You open the Object Spy by choosing **Tools** > **Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Methods Using the Object Spy" on page 108.

---

**To define a new test object property:**

1 In the object repository tree, select the test object for which you want to define a new property.

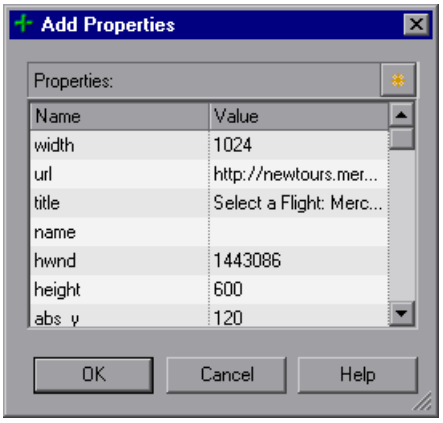
2 In the **Test object details** area, click the **Add description properties** button.

---

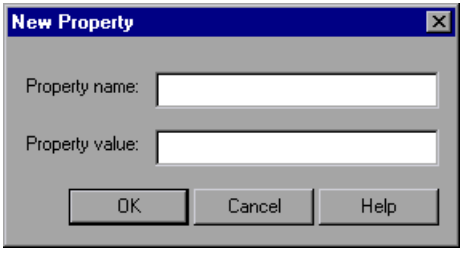
**Tip:** For a test object in the local object repository, you can also select the required test object and choose **Edit** > **Step Properties** > **Object Properties**, click the **Add description properties** button, and then perform the following steps in the Add Properties dialog box.

---

The Add Properties dialog box opens.




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




**4** Specify a valid test object property:

- **Property name**. Enter the property name.
- **Property value**. Enter the value for the property.

---

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

---

5 Click **OK** to add the property to the list and close the New Property dialog box. The new property is highlighted in the Add Properties dialog box.

6 Click **OK** while the new property is highlighted to include it in the object description and close the Add Properties dialog box.

## Removing Properties from a Test Object Description

You can remove properties from the description of a test object if you no longer want them to be part of the description. You can do this for test objects in the local object repository using the Object Repository window or Object Properties dialog box, and for test objects in the shared object repository using the Object Repository Manager.

**To remove a property from a test object description:**

1 In the object repository tree, select the test object whose description you want to modify.

2 In the **Test object details** area, select one or more properties that you want to remove from the test object description.

---

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

---

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

## Specifying Ordinal Identifiers

An ordinal identifier assigns a numerical value to a test object that indicates its order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). This ordered value provides a backup mechanism that enables QuickTest to create a unique description to recognize an object when the defined properties are not sufficient to do so. You can specify the ordinal identifier for test objects in the local object repository using the Object Repository window or Object Properties dialog box, and for test objects in the shared object repository using the Object Repository Manager.

For more information on ordinal identifiers, see "Selecting an Ordinal Identifier" on page 189.

**To specify an ordinal identifier:**

1 Select the test object whose ordinal identifier you want to specify.

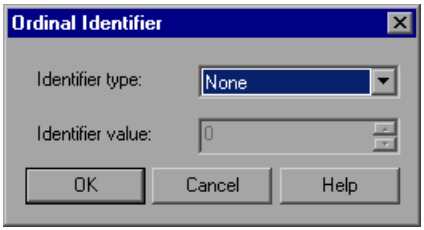
2 In the **Test object details** area, click in the cell to the right of the **Type, Value** cell under the **Ordinal identifier** row.

---

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

---

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



**4** In the **Identifier type** box, select one of the following options:

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

**5** In the **Identifier value** box, enter the numeric value of the ordinal identifier.

**6** Click **OK**. The ordinal identifier appears in the relevant row of the **Test object details** area for the selected object.

# Mapping Repository Parameter Values

You can map repository parameters that are used in shared object repositories that are associated with your component. Mapping a repository parameter to a value or parameter specifies the property values used to identify the test object during a run session. You can specify that the property value is taken from a constant value, or parameterize it using a local or component parameter.

You can map each repository parameter as required in each component that has an associated object repository containing repository parameters. For example, in one component you may want to retrieve the username object's text property value from an environment variable parameter, and in another component you may want the same object property value to use a constant value or a local parameter.

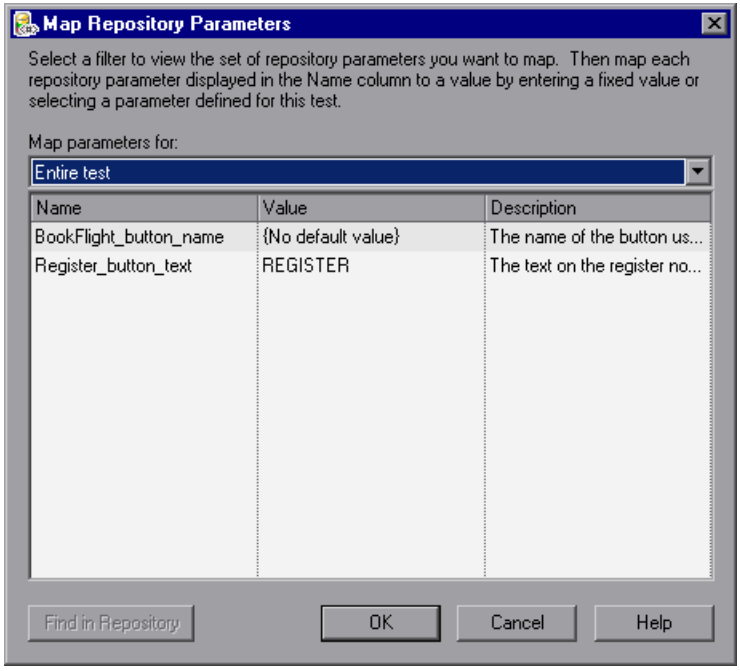
Before you map repository parameters, if you have more than one repository parameter with the same name in different shared object repositories that are associated with the same component, the repository parameter from the shared object repository with the highest priority (as defined in the shared object repositories list) is used. After you map repository parameters, QuickTest uses the mappings you defined. In addition, changing the priority or default values has no effect after the parameters are mapped.

When you open a component that uses an object repository with an object property value that is parameterized using a repository parameter with no default value, an indication that there is a repository parameter that needs mapping is displayed in the Missing Resources pane. You can then map the repository parameter as needed in the 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 component run. If the parameter is unmapped, meaning no default value was specified for it, the component run may fail if a test object cannot be identified because it has an unmapped parameter value.

**To map repository parameter values:**

**1** Choose **Resources** > **Map Repository Parameters**. The Map Repository Parameters dialog box opens.




---

**Tip:** If you have unmapped repository parameters (repository parameters without a default value) in your component, you can also open this dialog box by double-clicking the **Repository Parameters** row in the Missing Resources pane. For more information, see Chapter 28, "Handling Missing Resources."

---

The Map Repository Parameters dialog box contains the following options:

| Option name | Description |
|---|---|
| **Map parameters for** filter | Enables you to filter the list of parameters that is displayed. You can choose to display:<br>➤ **All unmapped parameters.** Displays all of the parameters in your test with unmapped values.<br>➤ **<Component name>.** (For example, LogIn) Displays all of the parameters in the specified component (with mapped or unmapped values). |
| **Name** column | The name of the repository parameter. |
| **Value** column | 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.<br>You can perform one of the following:<br>➤ Enter a new constant value.<br>➤ Parameterize the value by clicking in the **Value** cell of the relevant parameter and then clicking the parameterization button  .<br>➤ Reset a parameter to its default value by clicking in the **Value** cell of the relevant parameter and then clicking the **Reset to Default Value** button  . |
| **Description** column | A textual description of the parameter, if any. |
| **Find in Repository** button | Opens the Object Repository window and highlights the first test object in the object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so forth. |

---

**Note:** The repository parameter names, default values, and descriptions are defined in the Manage Repository Parameters dialog box. In addition, the names and descriptions can only be modified there. For more information, see "Managing Repository Parameters" on page 231.

---

**2** Click the **Map parameters for** arrow to select the list of parameter groups for which you want to define values. You can choose to display:

- **All unmapped parameters**. Displays all of the parameters in your test with unmapped values.
- **<Component name>**. (For example, LogIn) Displays all of the parameters in the specified component (with mapped or unmapped values).

**3** Click in the **Value** cell of the parameter you want to map. You can choose to map the value in one of the following ways:

- Enter a new constant value or modify an existing constant value by typing directly in the **Value** cell. You can also enter a constant value in the Value Configuration Options dialog box by clicking the parameterization button. For information on using this dialog box, see the *HP QuickTest Professional User's Guide*.

- Parameterize the value by clicking the parameterization button. The Value Configuration Options dialog box opens. You can parameterize the value using a local or component parameter. For information on using this dialog box, see the *HP QuickTest Professional User's Guide*.

- Restore the default value by clicking the **Clear Default Value** button. The default value, if any, that was defined in the Add Repository Parameter dialog box is displayed in the cell. For information on the Add Repository Parameter dialog box, see "Adding Repository Parameters" on page 233.

**4** Repeat step3 for any additional parameter values that you want to map. Then click **OK** to close the Map Repository Parameter dialog box.

# Adding Test Objects to an Object Repository

When you create a shared object repository for your keyword-driven testing infrastructure, you can add test objects to it in different ways. You can choose to add only a selected test object, or to add all test objects of a certain type, such as all button objects, or to add all test objects of a specific class, such as all WebButton objects. In addition, if you record a component, QuickTest adds each object on which you perform an operation to the local object repository (for objects that do not already exist in an associated shared object repository). You can also add test objects to the local object repository while editing your component.

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

---

**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 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 more information, see "Copying an Object to the Local Object Repository" on page 131.

---

If needed, you can merge test objects from the local object repository into a shared object repository. For more information, see Chapter 7, "Merging Shared Object Repositories."

You can also add test objects to a shared object repository while navigating through your application. For more information, see "Adding Test Objects Using the Navigate and Learn Option" on page 228.

---

**Tip:** You can also add a test object to the local object repository by choosing it from your application in the Select Object for Step dialog box (from a new step in the Keyword View). For more information, see "Selecting an Item for Your Step" on page 516.

---

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

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

---

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

---

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

**1** Perform one of the following:

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

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

QuickTest and the Object Repository window or Object Repository Manager are hidden, and the pointer changes into a pointing hand.

---

**Note:** If the window containing the object you want to add is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds. The window comes into the foreground. You can now point to and click the object you want. You can configure the length of time required to bring a window into the foreground in the General tab of the Options dialog box. For more information, see Chapter 20, "Setting Global Testing Options." You can also hold the left CTRL key to temporarily deactivate the pointing hand mechanism while you change the window focus. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

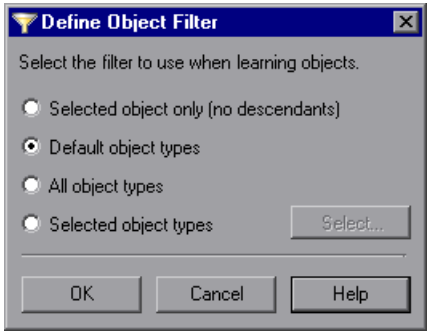
---

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

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

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

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




You can choose from the following options:

- **Selected object only (no descendants).** Adds to the object repository the previously selected object's properties and values, without its descendant objects.
- **Default object types.** Adds to the object repository the previously selected object's properties and values, with the properties and values of its descendant objects according to the object types specified by the default filter. You can see which objects are in the default filter by clicking the **Select** button and then clicking the **Default** button.

- **All object types.** Adds to the object repository the previously selected object's properties and values, together with the properties and values of all of its descendant objects.
- **Selected object types.** Adds to the object repository the previously selected object's properties and values, as well as the properties and values of its descendant objects according to the object types and classes you specify in the object filter. You specify the objects and classes in the filter by clicking the **Select** button and selecting the required items in the Select Object Types dialog box. For more information on the Select Object Types dialog box, see "Understanding the Select Object Types Dialog Box" on page 162.

**4** Select the required option and click **OK** to close the Define Object Filter dialog box and add the specified objects to the object repository according to the selected object filter.

**5** The Object Repository window is redisplayed, showing the new local objects and their properties and values in the object repository. If you chose to add the objects from the Object Repository Manager, the objects are added to the active shared object repository.

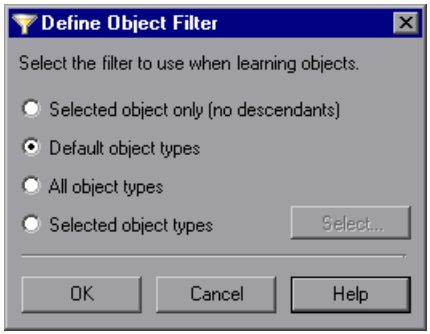
QuickTest also adds the new object's parent objects if they do not already exist in the object repository. Local objects are shown in black in the object repository tree to indicate they are editable; shared objects are shown in gray and can only be edited in the Object Repository Manager.

You can edit the new test object's details just as you would edit any other object in a local or shared object repository. For more information, see "Viewing and Modifying Test Object Properties" on page 130.

## Understanding the Define Object Filter Dialog Box

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

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




You can choose from the following options:

- **Selected object only (no descendants)**. Adds to the object repository the previously selected object's properties and values, without its descendant objects.
- **Default object types**. Adds to the object repository the previously selected object's properties and values, with the properties and values of its descendant objects according to the object types specified by the default filter. You can see which objects are in the default filter by clicking the **Select** button and then clicking the **Default** button.

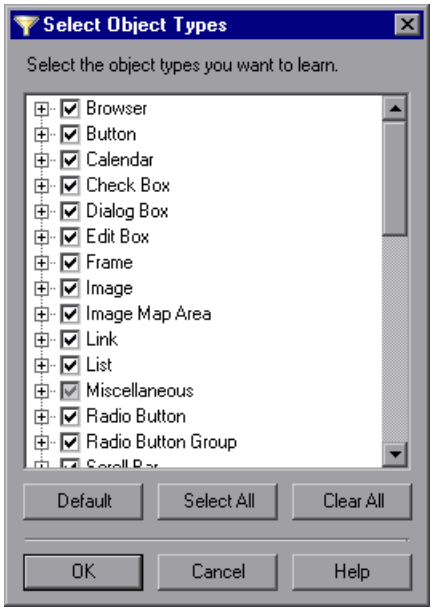
- **All object types**. Adds to the object repository the previously selected object's properties and values, together with the properties and values of all of its descendant objects.
- **Selected object types**. Adds to the object repository the previously selected object's properties and values, as well as the properties and values of its descendant objects according to the object types and classes you specify in the object filter. You specify the objects and classes in the filter by clicking the **Select** button and selecting the required items in the Select Object Types dialog box. For more information on the Select Object Types dialog box, see "Understanding the Select Object Types Dialog Box" on page 162.

## Understanding the Select Object Types Dialog Box

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

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

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




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

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

---

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

---

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

## Defining New Test Objects

You can define test objects in your object repository that do not yet exist in your application. This enables you to prepare an object repository and build 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.

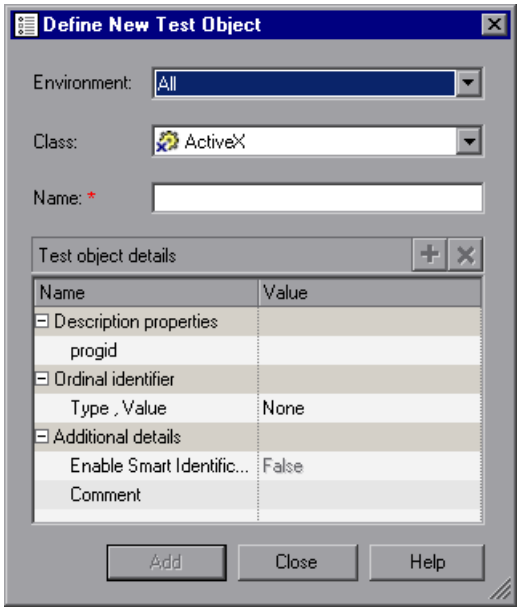
When you define a new object in the object repository as described in this section, the object is added to the local object repository and can only be used by the current component. If you want to add the object to the shared object repository so that it can be used in multiple components, you must add it using the Object Repository Manager. For more information, see Chapter 6, “Managing Object Repositories.”

After you have defined the new test object, if the properties of the object in your application do not match the test object description that you defined, or if an object has been updated in your application, you can update the object description at any time. For more information, see “Updating Test Object Properties from an Object in Your Application” on page 138.

**To define a new test object:**

1 Select the object under which you want to define the new object, according to the correct object hierarchy.

2 Click the **Define New Test Object** button or choose **Object** > **Define New Test Object**. The Define New Test Object dialog box opens.




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

---

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

---

4 In the **Class** box, select the class of the test object you want to define.

5 In the **Name** box, enter a name for the new test object. After you enter a name, the **Test object details** area is enabled.

6 In the **Test object details** area, define the properties and values for your test object. The **Test object details** area automatically contains the mandatory properties defined for the object class in the Object Identification dialog box. You can add or remove properties as required, and define values for the properties. For more information, see "Viewing and Modifying Test Object Properties" on page 130.

7 Click **Add**. The new test object is added to the local object repository in the selected location.

8 Repeat step3 to step7 to define additional test objects, or click **Close** to close the Define New Test Object dialog box.

## Copying, Pasting, and Moving Objects in the Object Repository

You can copy, paste, and move test objects and checkpoint and output objects in the local object repository using the Object Repository window, and copy, paste, and move objects both within a shared object repository and between shared object repositories using the Object Repository Manager. You can also copy objects from a shared object repository to the local object repository to modify them locally. For more information, see "Copying an Object to the Local Object Repository" on page 131.

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

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

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

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

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

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

**To move or copy an object without its child objects:**

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

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

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

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

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

**To copy objects from one shared object repository to another:**

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

**To move objects from one shared object repository to another:**

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

## Guidelines for Copying, Pasting, and Moving Objects

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

- You cannot modify the root node of an object repository.
- If you change the object hierarchy, ensure that the new hierarchy is valid.
- If you paste or move an object to a different hierarchical level, you can choose whether to copy all objects up to the shared parent object (in the message displayed when you perform such an operation).
- In the Object Repository window, when you copy, paste, and move objects from a shared object repository associated with a component, the objects are copied, pasted, or moved to the local object repository of the component.
- If you move an object to its immediate parent, QuickTest creates a copy of the object (renamed with an incremental suffix) and pastes it as a sibling of the original object.
- If you cut or copy an object, and then paste it on its parent object, QuickTest creates copy of the object (renamed with an incremental suffix) and inserts it at the same level as the original object.
- You cannot move an object to any of its descendants.
- You cannot copy or move an object to be a child of a bottom-level object (an object that cannot contain a child object) in the object hierarchy.
- You cannot copy, paste, or move objects that have unmapped repository parameters from a shared object repository to the local object repository. If you copy, paste, or move an object from a shared object repository to the local object repository and the object or one of its parent objects are parameterized using one or more repository parameters, the repository parameter values are converted when you copy, paste, or move the object. For example, if the repository parameter is mapped to a local parameter, the property is parameterized using a local parameter. If the value is a constant value, the property receives the same constant value.

# Deleting Objects from the Object Repository

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

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

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

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

---

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

---

**To delete an object from the object repository:**

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

---

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

---

# Locating Objects

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

## Finding Objects in an Object Repository

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

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

---

**Notes:**

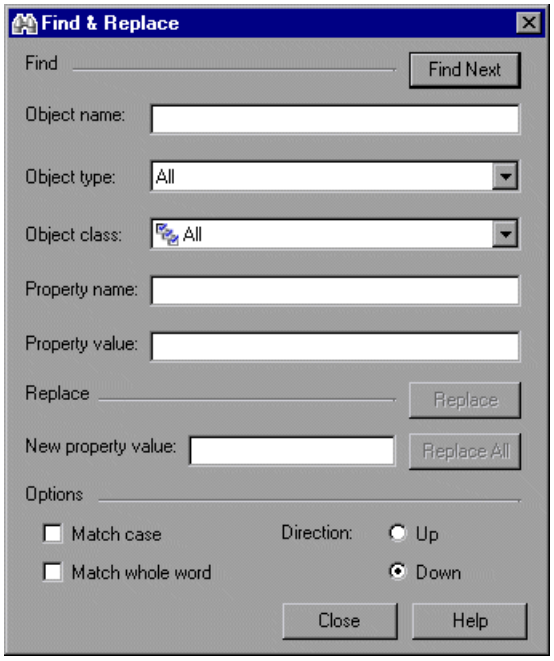
- The Find and Replace dialog box can only find checkpoint and output values by searching for the object name.
- You cannot use the Find and Replace dialog box to replace property or object names. You cannot replace property values in a read-only component.

---

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

1 Make sure that the relevant object repository is open (in the Object Repository window or Object Repository Manager).

2 Click the **Find & Replace** button or choose **Edit** > **Find & Replace**. The Find & Replace dialog box opens.



**3** Specify one or more criteria by which you want to search for the object, property, or property value:

- **Object name.** Enter the name or partial name of the object you want to find.
- **Object type.** Select the type of object you want to find, for example, **Button**.

---

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

---

- **Object class.** Select the class of object you want to find, for example, **WebButton**. The classes available depend on the selection you made in the **Object type** box.
- **Property name.** Specify the name or partial name of the property you want to find.
- **Property value.** Specify the property value or partial property value you want to find.

**4** If you specified a property value and want to replace it with a different value, enter the new property value in the **New property value** box.

**5** Specify the search parameters, as follows:

- If you want the search to distinguish between upper and lower case letters, select **Match case**.
- If you want the search to find only complete words that exactly match the single word you entered, select **Match whole word**.
- Specify the direction in which you want to search: **Up** or **Down**.

**6** Perform the find or replace operation in one of the following ways. The search is performed on the entire object repository, starting with the currently selected object and in the direction you specified. To find the next instance, click **Find Next** again.

- To find the specified object, property, or property value, click **Find Next**. The first instance of the searched word is displayed.
- To individually find and replace each instance of the property value for which you are searching, click **Find Next**. When an instance is found, click **Replace**. The property value is replaced, and the next instance of the property value, if any, is highlighted.
- To replace all instances of the specified property value with the new property value, click **Replace All**. Instances in shared object repositories that are not editable are not changed.

## Highlighting an Object in Your Application

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

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

---

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

---

**To highlight an object in your application:**

1 Make sure your application is open to the correct window or page.

2 Select the test object you want to highlight in your object repository.

3 Click the **Highlight in Application** button or choose **View** > **Highlight in Application**. The selected object is highlighted with a border in the application.


---

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

---

## Locating a Test Object in the Object Repository

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

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

**To locate an object in the object repository:**

1 Make sure your application is open to the correct window or page.

2 Click the **Locate in Repository** button or choose **View** > **Locate in Repository**. QuickTest is hidden, and the pointer changes into a pointing hand.

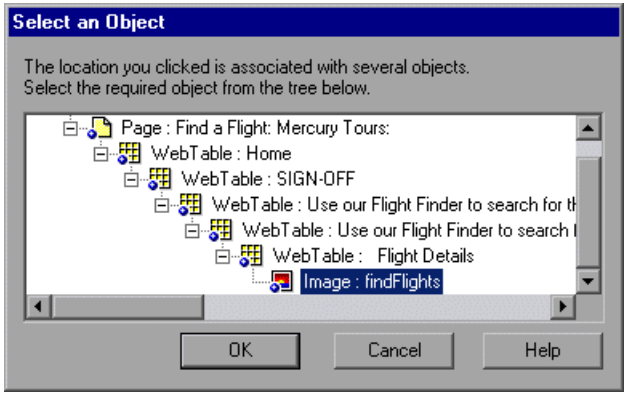
**3** Use the pointing hand to click on the required object in your application.

---

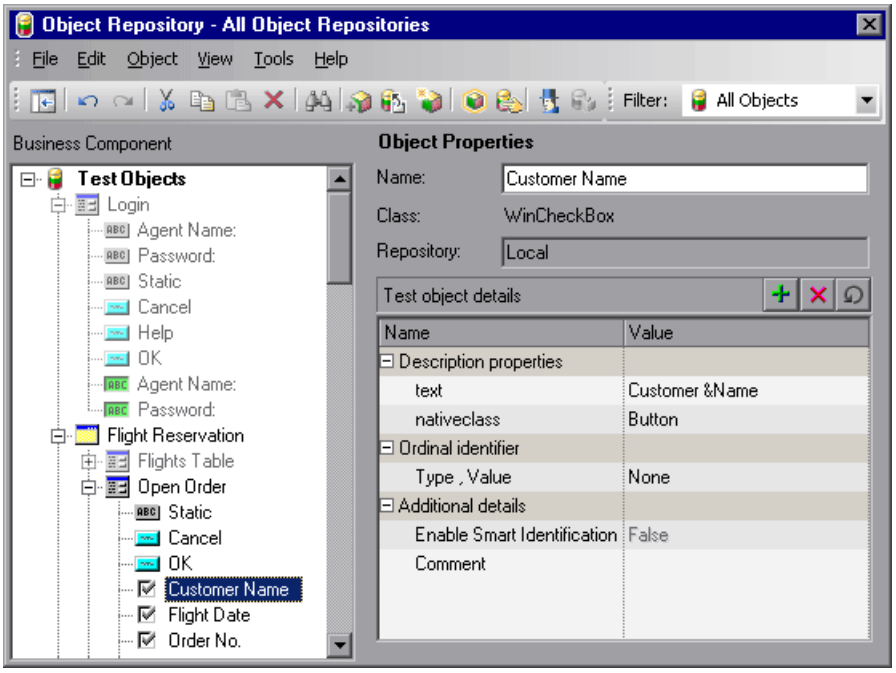
**Tip:** You can hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to click is partially hidden by another window, you can also hold the pointing hand over the partially hidden window for a few seconds until the window comes into the foreground and you can point and click on the object you want. Additionally, 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 location you clicked is associated with more than one object, the Select an Object dialog box opens.



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




---

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

---

# Working with Test Objects During a Run Session

The first time QuickTest encounters an object during a run session, it creates a temporary version of the test object for that run session. QuickTest uses the object description to create this temporary version of the object. For the remainder of the component, QuickTest refers to the temporary version of the test object rather than to the test object in the object repository.

---

**Note:** The Object Repository window is read-only during record and run sessions.

---

## Creating Test Objects During a Run Session

You can use programmatic descriptions to create temporary versions of test objects that represent objects from your application. You can perform operations on those objects without referring to the object repository. For example, suppose an edit box was added to a form on your Web site. You can use a programmatic description to add a statement in a user-defined function that enters a value in the new edit box. QuickTest could then identify the object even though the object was never added to the object repository. For more information on programmatic descriptions, see "Using Programmatic Descriptions" on page 332.

## Modifying Test Object Properties During a Run Session

You can modify the properties of the temporary version of the object during the run session without affecting the permanent values in the object repository by adding a SetTOProperty statement in a user-defined function.

Use the following syntax for the SetTOProperty method:

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

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

# Exporting Local Objects to a Shared Object Repository

You can export all of the test objects, checkpoint objects, and output value objects contained in a component's local object repository to a new shared object repository in the file system or to a Quality Center project (if QuickTest is connected to Quality Center). This enables you to make the local objects accessible to other components. You export local objects to a new shared object repository using the Object Repository window.

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 component using the Map Repository Parameters dialog box (described in "Mapping Repository Parameter Values" on page 152). For more information on repository parameters, see Chapter 6, "Managing Object Repositories."

---

**Tip:** After you export the local objects, you can use the Object Repository Merge Tool to merge the test objects from the shared object repository containing the exported objects with another shared object repository. For more information, see Chapter 7, "Merging Shared Object Repositories."

---

**To export local objects to a new shared object repository:**

1. Open the component that has the local objects you want to export.
2. Open the Object Repository window by selecting **Resources** > **Object Repository** or clicking the **Object Repository** button.

3. Select **File** > **Export Local Objects**. The Export Object Repository dialog box opens.

---

**Note:** If you are connected to Quality Center, the dialog box that opens is different from the standard file system dialog box. You can switch between the two dialog box versions by clicking the **File System** and **Quality Center** buttons in the Export Object Repository dialog box.

---

**4** Select the location in which to save the file, specify the **File name** or **Attachment Name**, and click **Save** or **OK** (depending on whether you are saving it to the file system or a Quality Center project).

If you chose **Export Local Objects**, the local objects are exported to the specified shared object repository (a file with a **.tsr** extension). Your 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.

# 5

# Configuring Object Identification

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

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

**This chapter includes:**



## About Configuring Object Identification

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

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

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

---

**Note:** If the combination of all defined mandatory and assistive properties is not sufficient to create a unique test object description, QuickTest also learns the value for the selected ordinal identifier. For more information, see "Selecting an Ordinal Identifier" on page 189.

---

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

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

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

# Understanding the Object Identification Dialog Box

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

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

---

**Notes:**

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

---

For more information, see:


- "Configuring Mandatory and Assistive Properties" on page 184
- "Selecting an Ordinal Identifier" on page 189
- "Enabling and Disabling Smart Identification" on page 194
- "Restoring Default Object Identification Settings for Test Objects" on page 195
- "Generating Automation Scripts for Your Object Identification Settings" on page 195

## Configuring Mandatory and Assistive Properties

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

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

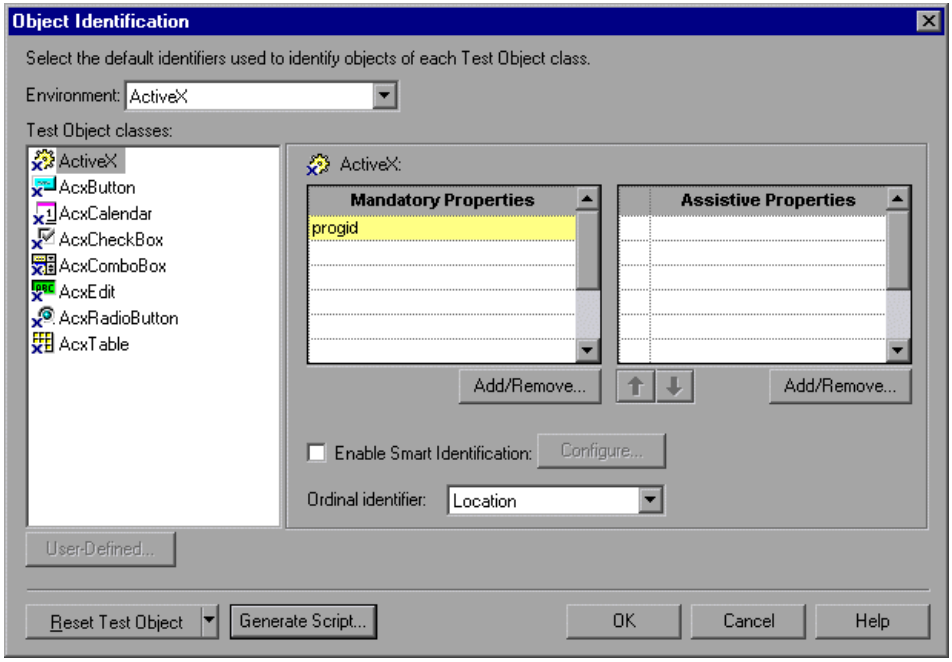
For example, the default mandatory properties for a Web Image object are the **alt**, **html tag**, and **image type** properties. There are no default assistive properties defined. Suppose your Web site contains several space holders for different collections of rotating advertisements. You want to create a 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 component, and most likely another **alt** value will be captured when you run the 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 QuickTest to uniquely identify the object.

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

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

**1** Choose **Tools** > **Object Identification**. The Object Identification dialog box opens.




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

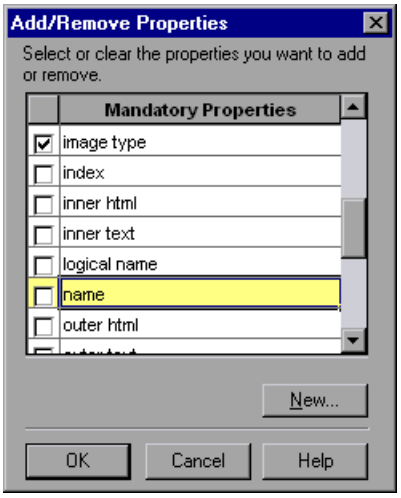
---

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

---

**3** In the **Test Object classes** list, select the test object class you want to configure.

**4** In the **Mandatory Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for mandatory properties opens.




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

---

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

---

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

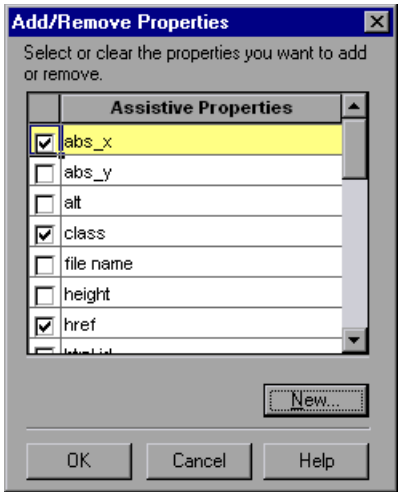
---

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

---

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

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




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

---

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

---

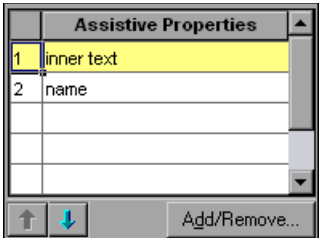
You can specify a new property by clicking **New** and specifying a valid property name in the displayed dialog box.

---

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

---

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




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

## Selecting an Ordinal Identifier

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

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

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

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

- **Index.** Indicates the order in which the object appears in the application code relative to other objects with an otherwise identical description. For more information, see "Identifying an Object Using the Index Property" on page 190.
- **Location.** Indicates the order in which the object appears within the parent window, frame, or dialog box relative to other objects with an otherwise identical description. For more information, see "Identifying an Object Using the Location Property" on page 191.
- **CreationTime.** (Browser object only.) Indicates the order in which the browser was opened relative to other open browsers with an otherwise identical description. For more information, see "Identifying an Object Using the CreationTime Property" on page 192.

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


---

**Tip:** While recording, if QuickTest successfully creates a unique test object description using the mandatory and assistive properties, it does not learn an ordinal identifier value. You can add an ordinal identifier to an object's test object properties at a later time using the **Add/Remove** option from the Object Properties or Object Repository dialog box. For more information, see Chapter 4, "Working with Objects."

---

### Identifying an Object Using the Index Property

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

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

For example, suppose a page contains the following objects:

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

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

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

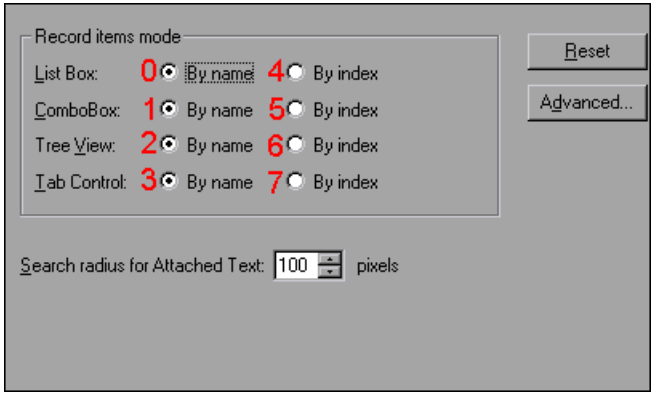
In contrast, the following statement refers to the second item in the list, as that is the first object of any type (WebElement) with the name UserName:

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

## Identifying an Object Using the Location Property

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

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



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

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

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

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

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

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

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

### Identifying an Object Using the CreationTime Property

While learning a browser object, if QuickTest is unable to uniquely identify the object according to its test object description, it assigns a value to the **CreationTime** test object property. This value indicates the order in which the browser was opened relative to other open browsers with an otherwise identical description. The first browser that opens receives the value CreationTime = 0.

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

For example, if you record a component on three otherwise identical browsers that are opened at 9:01 pm, 9:03 pm, and 9:05 pm, QuickTest assigns the CreationTime values, as follows: CreationTime = 0 to the 9:01 am browser, CreationTime = 1 to the 9:03 am browser, and CreationTime = 2 to the 9:06 am browser.

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

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

If a step was recorded on a browser 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 recorded on a browser with CreationTime = 6, but during the run session there are only two open browsers, with CreationTime = 0 and CreationTime = 1, then the step runs on the last browser opened, which in this example is the browser with CreationTime = 1.

---

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

---

## Enabling and Disabling Smart Identification

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

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

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

---

**Note:** Even if you choose to learn Smart Identification properties for an object, you can disable use of the Smart Identification mechanism for a specific object in the Object Properties or Object Repository dialog box. For more information, see Chapter 4, "Working with Objects."

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

---

For more information on the Smart Identification mechanism, see "Configuring Smart Identification" on page 196.

## Restoring Default Object Identification Settings for Test Objects

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

Only built-in object properties can be reset. When you reset the settings for the Standard Windows environment, user-defined objects are also deleted. For more information on user-defined objects, see "Mapping User-Defined Test Object Classes" on page 206.

---

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

---

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

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

## Generating Automation Scripts for Your Object Identification Settings

You can click the **Generate Script** button to generate an automation script containing the current object identification settings. For more information, see "Automating QuickTest Operations" on page 817, or see the *QuickTest Automation Reference* (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Automation**).

# Configuring Smart Identification

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

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

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

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

The Smart Identification mechanism uses two types of properties:

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

### Understanding the Smart Identification Process

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

**1** QuickTest "forgets" the learned test object description and creates a new **object candidate** list containing the objects (within the object's parent object) that match all of the properties defined in the Base Filter Properties list.

**2** QuickTest filters out any object in the object candidate list that does not match the first property listed in the Optional Filter Properties list. The remaining objects become the new object candidate list.

**3** QuickTest evaluates the new object candidate list:

- ➤ If the new object candidate list still has more than one object, QuickTest uses the new (smaller) object candidate list to repeat step 2 for the next optional filter property in the list.
- ➤ If the new object candidate list is empty, QuickTest ignores this optional filter property, returns to the previous object candidate list, and repeats step 2 for the next optional filter property in the list.
- ➤ If the object candidate list contains exactly one object, then QuickTest concludes that it has identified the object and performs the statement containing the object.

**4** QuickTest continues the process described in steps 2 and 3 until it either identifies one object, or runs out of optional filter properties to use.

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

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

### Reviewing Smart Identification Information in the Test Results

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

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

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

For more information, see "Analyzing Smart Identification Information in the Test Results" on page 665.

### Walking Through a Smart Identification Example

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

Suppose you have the following statement in your component:

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

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

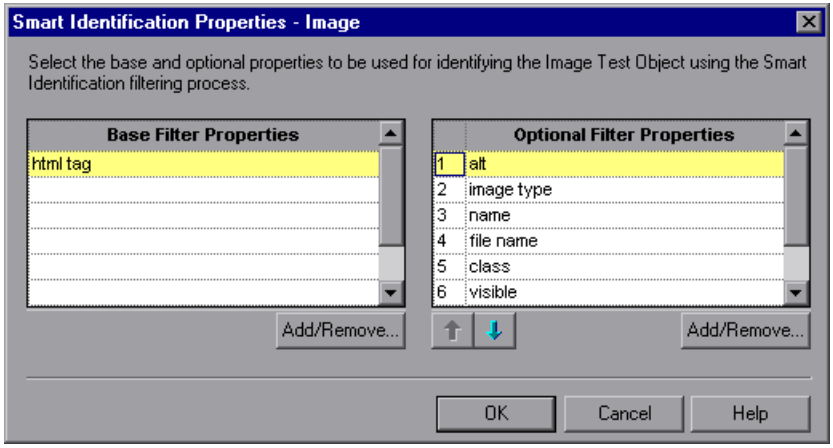
| Name | Value |
|---|---|
| ⊟ Description properties | |
| image type | Image Button |
| html tag | INPUT |
| alt | Login |

However, at some point after you created your 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 component, QuickTest is unable to identify the Login button based on the learned description. However, QuickTest succeeds in identifying the Login button using its Smart Identification definition.

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

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




The learned values are as follows:

**Base Filter Properties:**

| Property | Value |
|---|---|
| **html tag** | INPUT |

**Optional Filter Properties:**

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

**2** QuickTest begins the Smart Identification process by identifying the five objects on the Mercury Tours page that match the base filter properties definition (**html tag** = INPUT). QuickTest considers these to be the object candidates and begins checking the object candidates against the **Optional Filter Properties** list.

**3** QuickTest checks the **alt** property of each of the object candidates, but none have the **alt** value: Login, so QuickTest ignores this property and moves on to the next one.

**4** QuickTest checks the **image type** property of the each of the object candidates, but none have the **image type** value: Image Button, so QuickTest ignores this property and moves on to the next one.

**5** QuickTest checks the **name** property of each of the object candidates, and finds that two of the objects (both the basic and VIP Login buttons) have the name: login. QuickTest filters out the other three objects from the list, and these two login buttons become the new object candidates.

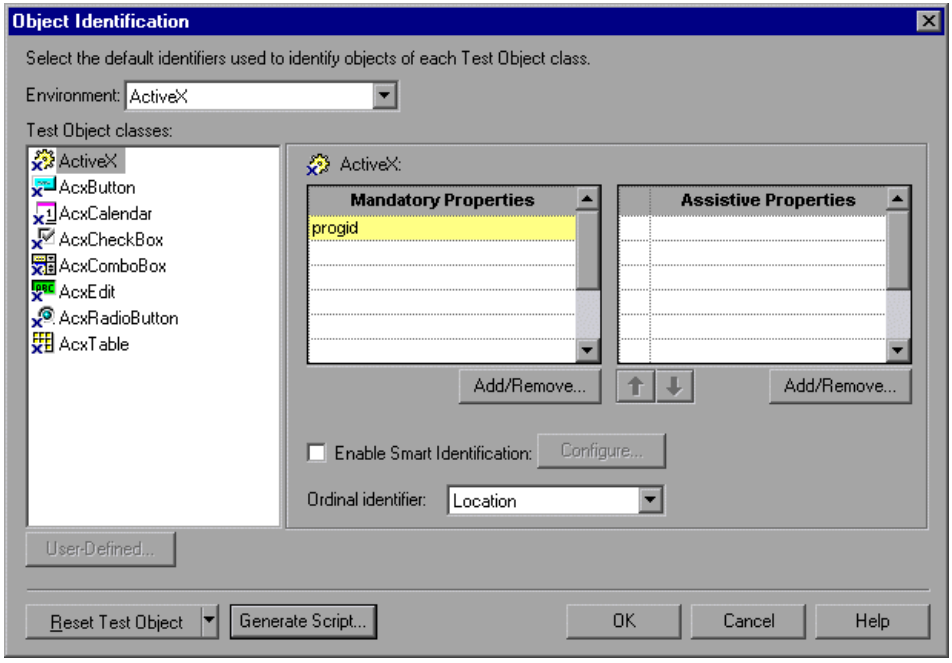
**6** QuickTest checks the **file name** property of the two remaining object candidates. Only one of them has the file name login.gif, so QuickTest correctly concludes that it has found the Login button and clicks it.

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

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

### To configure Smart Identification properties:

**1** Choose **Tools** > **Object Identification**. The Object Identification dialog box opens.



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

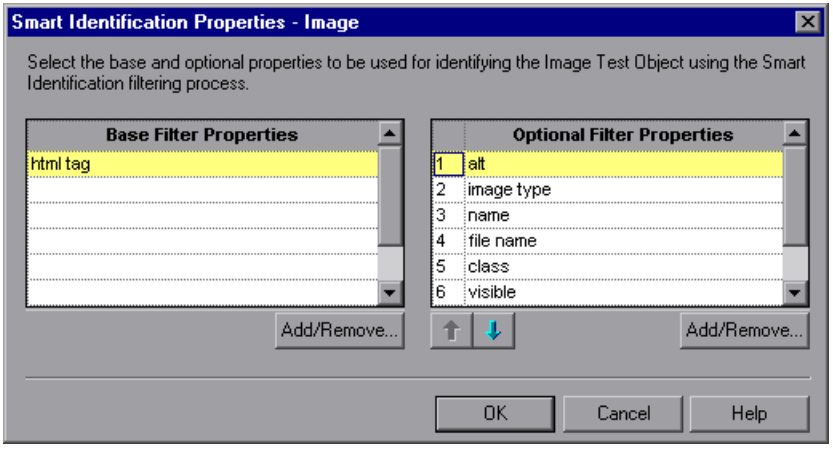
---

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

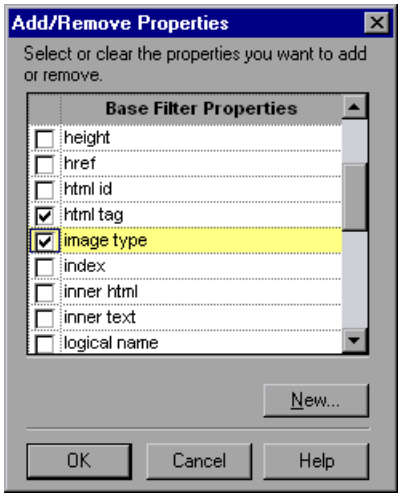
---

**3** Select the test object class you want to configure.

**4** Click the **Configure** button next to the **Enable Smart Identification** check box. The **Configure** button is enabled only when the **Enable Smart Identification** option is selected. The Smart Identification Properties dialog box opens.



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




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

---

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

---

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

---

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

---

**7** Click **OK** to close the Add/Remove Properties dialog box. The updated set of base filter properties is displayed in the **Base Filter Properties** list.

**8** In the **Optional Filter Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for optional filter properties opens.


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

---

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

---

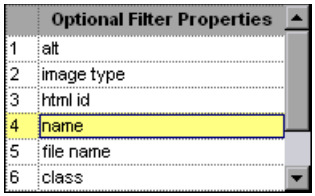
You can specify a new property by clicking **New** and specifying a valid property name in the displayed dialog box.

---

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

---

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




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

# Mapping User-Defined Test Object Classes

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

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

---

**Notes:**

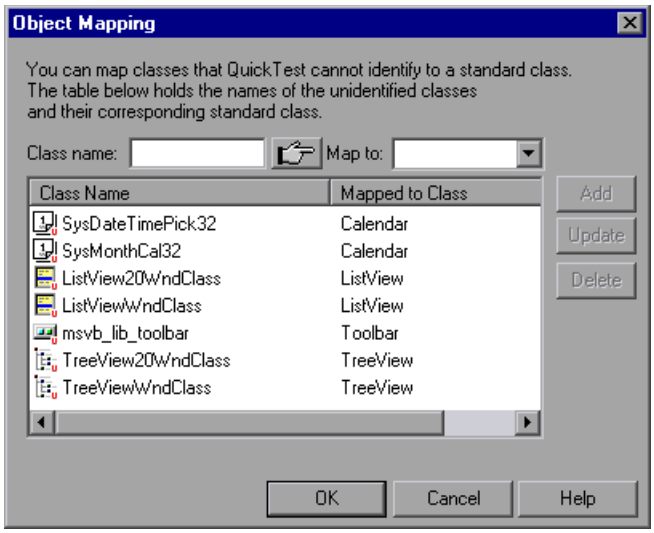
- You can define user-defined classes only when **Standard Windows** is selected in the **Environment** box.
- If you click the down arrow on the **Reset Test Object** button and select **Reset Environment**, when **Standard Windows** is selected in the **Environment** box, all of the user-defined test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

---

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

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

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




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

---

**Tip:** Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

---

**5** In the **Map to** box, select the standard object class to which you want to map your user-defined object class and click **Add**. The class name and mapping is added to the object mapping list.

**6** If you want to map additional objects to standard classes, repeat steps 4-5 for each object.

7 Click **OK**. The Object Mapping dialog box closes and your object is added to the list of Standard Windows test object classes as a user-defined test object. Note that your object has an icon with a red U in the lower-right corner, identifying it as a user-defined class.

8 Configure the object identification settings for your user defined object class just as you would any other object class. For more information, see "Configuring Mandatory and Assistive Properties" on page 184, and "Configuring Smart Identification" on page 196.

**To modify an existing mapping:**

1 In the Object Mapping dialog box, select the class you want to modify from the object mapping list. The class name and current mapping are displayed in the Class name and Map to boxes.

2 Select the standard object class to which you want to map the selected user-defined class and click **Update**. The class name and mapping is updated in the object mapping list.

3 Click **OK** to close the Object Mapping dialog box.

**To delete an existing mapping:**

1 In the Object Mapping dialog box, select the class you want to delete from the object mapping list.

2 Click **Delete**. The class name and mapping is deleted from the object mapping list in the Object Mapping dialog box.

3 Click **OK**. The Object Mapping dialog box closes and the class name is deleted from the Standard Windows test object classes list in the Object Identification dialog box.

# 6

# Managing Object Repositories

The Object Repository Manager enables you to manage all of the shared object repositories used in your organization from a single, central location, including adding and defining objects, modifying objects and their descriptions, parameterizing repositories to make them more generic, maintaining and organizing repositories, merging repositories, and importing and exporting repositories in XML format.

**This chapter includes:**

# About Managing Object Repositories

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

Each object repository contains the information that enables QuickTest to identify the objects in your application. QuickTest enables you to maintain the reusability of your components by storing all the information regarding your test objects in a shared object repository. When objects in your application change, the Object Repository Manager provides a single, central location in which you can update test object information for multiple components.

---

**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 component. For more information on local object repositories, see Chapter 4, "Working with Objects."

---

If an object with the same name and description is located in both the local object repository and in a shared object repository that is associated with the same component, the component uses the local object definition. If an object with the same name and description is located in more than one shared object repository, and these shared object repositories are all associated with the same component, QuickTest uses the object definition from the first occurrence of the object, according to the order in which the shared object repositories are associated with the component. For more information on associating shared object repositories, see "Managing Shared Object Repositories" on page 432.

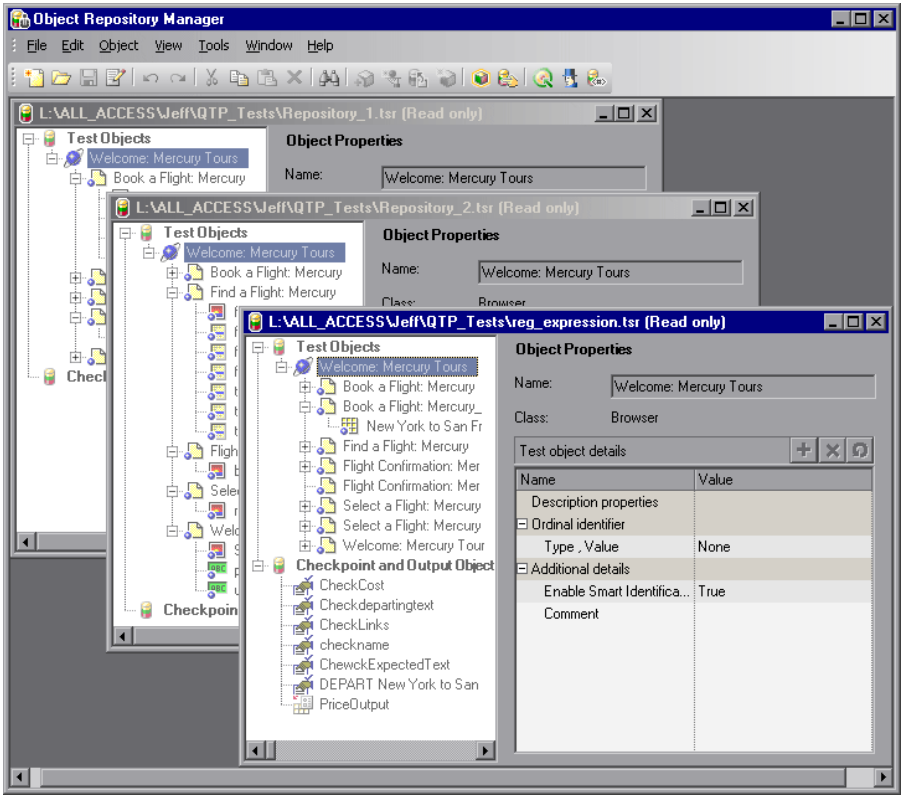
You can use the same shared object repository with multiple components. You can also use multiple object repositories with each component. In addition, you can save objects directly with an component in a local object repository. This enables them to be accessed only from that component.

If one or more of the property values of an object in your application differ from the property values QuickTest uses to identify the object, your component may fail. Therefore, when the property values of objects in your application change, you should modify the corresponding test object property values in the corresponding object repository so that you can continue to use your existing components.

You can modify objects in a shared object repository using the Object Repository Manager, as described in this chapter. You can modify objects stored in a local object repository using the Object Repository window. For information on the Object Repository window, see Chapter 4, "Working with Objects."

# Understanding the Object Repository Manager

You open the Object Repository Manager by choosing **Resources** > **Object Repository Manager**. The Object Repository Manager enables you to open multiple shared object repositories and modify them as needed. You can open shared object repositories both from the file system and from a Quality Center project.




---

**Tip:** While the Object Repository Manager is open, you can continue working with other QuickTest windows.

---

You can open as many shared object repositories as you want. Each shared object repository opens in a separate document window. You can then resize, maximize, or minimize the windows to arrange them as you require to copy, drag, and move objects between different shared object repositories, as well as perform operations on a single object repository. For more information on the details shown in the shared object repository windows, see "Understanding the Shared Object Repository Windows" on page 217.

You open shared object repositories from the Open Shared Object Repository dialog box. In this dialog box, the **Open in read-only mode** check box is selected, by default. If you clear this check box, the shared object repository opens in editable mode. Otherwise, the shared object repository opens in read-only mode and you must click the **Enable Editing** button to modify it. For more information, see "Editing Object Repositories" on page 226.

When you choose a menu item or click a toolbar button in the Object Repository Manager, the operation you select is performed on the shared object repository whose window is currently active (in focus). The name and file path of the shared object repository is shown in the title bar of the window. For more information on the Object Repository Manager toolbar buttons, see "Using the Object Repository Manager Toolbar" on page 214.

Many of the shared object repository operations you can perform in the Object Repository Manager are done in a similar way to how you modify objects stored in a local object repository (using the Object Repository window). For this reason, many of the procedures are actually described in Chapter 4, "Working with Objects." Most of the procedures apply equally to the Object Repository Manager and the Object Repository window, but the windows and options may differ slightly.

## Using the Object Repository Manager Toolbar

You can access frequently performed operations using the Object Repository Manager toolbar. The Object Repository Manager toolbar contains the following buttons:

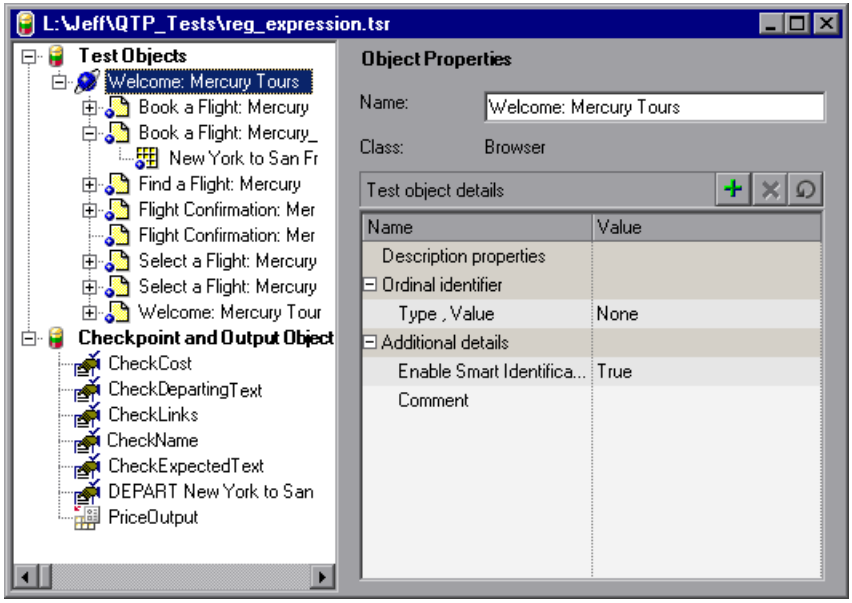
| Button | Description |
|---|---|
|  | Enables you to create a new shared object repository. For more information, see "Creating New Object Repositories" on page 219. |
|  | Enables you to open a shared object repository from the file system or from Quality Center. For more information, see "Opening Object Repositories" on page 219. |
|  | Enables you to save the active shared object repository to the file system or to Quality Center. For more information, see "Saving Object Repositories" on page 221. |
|  | Enables you to edit the active shared object repository, by making the shared object repository editable. For more information, see "Editing Object Repositories" on page 226. |
|  | Enables you to undo the previous operation performed in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 166. |
|  | Enables you to redo the operation that was previously undone in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 166. |
|  | Enables you to cut the selected item or object in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 166. |
|  | Enables you to copy the selected item or object to the Clipboard in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 166. |

| Button | Description |
|---|---|
|  | Enables you to paste the data from the Clipboard to the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Copying, Pasting, and Moving Objects in the Object Repository” on page 166. |
|  | Enables you to delete the selected item or object in the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Deleting Objects from the Object Repository” on page 169. |
|  | Enables you to find an object, property, or property value in the active shared object repository. You can also find and replace specified property values. You do this in the same way as in a local object repository. For more information, see “Finding Objects in an Object Repository” on page 170. |
|  | Enables you to add objects to the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Adding Test Objects to an Object Repository” on page 156. |
|  | Enables you to update test object properties in the active shared object repository according to the actual properties of the object in your application. You do this in the same way as in a local object repository. For more information, see “Updating Test Object Properties from an Object in Your Application” on page 138. |
|  | Enables you to define a test object that does not yet exist in your application and add it to the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Defining New Test Objects” on page 164. |
|  | Enables you to select an object in the active shared object repository and highlight it in your application. You do this in the same way as in a local object repository. For more information, see “Highlighting an Object in Your Application” on page 173. |
|  | Enables you to select an object in your application and highlight it in the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Locating a Test Object in the Object Repository” on page 174. |

| Button | Description |
|---|---|
|  | Enables you to connect to Quality Center to work with object repository files stored in a Quality Center project. You can connect to Quality Center from the main QuickTest window or from the Object Repository Manager. For more information, see “Connecting to Your Quality Center Project” on page 44. |
|  | Enables you to open the Object Spy to view run-time or test object properties and values of objects in your application. For more information, see “Viewing Object Properties and Methods Using the Object Spy” on page 108. |
|  | Enables you to add, edit, and delete repository parameters in the active shared object repository. For more information, see “Managing Repository Parameters” on page 231. |

## Understanding the Shared Object Repository Windows

Each shared object repository that you open in the Object Repository Manager is displayed in a standalone document window. Each shared object repository window displays a tree of all the objects in the object repository, together with object information for the selected object.




For each object you select in the tree, the Object Repository window displays information about the selected object. You can view the object description of any object in the shared object repository, modify objects and their properties, and add test objects to the shared object repository.

**Notes:**

- You cannot add checkpoint or output value objects to a shared object repository via the Object Repository Manager.
- Test objects of environments that are not installed with QuickTest are displayed with an unknown icon (question mark) in the test object tree.

For more information, see "Managing Objects in Shared Object Repositories" on page 224 and "Modifying Object Details" on page 235.

Each object repository window contains the following information:

| Information | Description |
|---|---|
| **Object Repository tree** | Contains all objects in the shared object repository. |
| **Name** | Specifies the name that QuickTest assigns to the selected object. You can change the object name. For more information, see "Renaming Test Objects" on page 140. |
| **Class** | Specifies the class of the selected object. |
| Object details | Enables you to view the properties and property values used to identify a test object during a run session or the properties of a checkpoint or output object. For more information, see "Modifying Object Details" on page 235. |

**Note:** Even when steps containing an object are deleted from your component, the objects remain in the object repository. You can delete objects from a shared object repository using the Object Repository Manager, in much the same was as you delete objects from a local object repository. For more information, see "Deleting Objects from the Object Repository" on page 169.

# Working with Object Repositories

You can use the Object Repository Manager to create new object repositories, open and modify existing object repositories, and save and close them when you are finished.

## Creating New Object Repositories

You can create a new object repository, add objects to it, and then save it. You can then associate one or more components with the object repository from within QuickTest. For more information on associating shared object repositories, see "Managing Shared Object Repositories" on page 432.

**To create a new object repository:**

In the Object Repository Manager, choose **File** > **New** or click the **New** button. A new object repository opens. You can now add objects to it, modify it, and save it. For more information, see "Managing Objects in Shared Object Repositories" on page 224 and "Saving Object Repositories" on page 221.

## Opening Object Repositories

You can open existing object repositories to view or modify them. You can open object repositories from the file system or from a Quality Center project.

You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File** > **Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see "Connecting to Your Quality Center Project" on page 44.

**Note for users of previous QuickTest versions:**

When you open an object repository that was created using a version of QuickTest earlier than version 9.0, QuickTest converts it to the current format when you make it editable.

If the object repository contains test objects from add-ins, the relevant add-in must be installed to convert the object repository to the current format. Otherwise, you can open it only in read-only format.

If you do not want to convert the object repository, you can view it in read-only format. After the file is converted and you save it, you cannot use it with earlier versions of QuickTest.

**To open an object repository:**

**1** In the Object Repository Manager, choose **File** > **Open** or click the **Open** button. The Open Shared Object Repository dialog box opens.

**Note:** If you are connected to Quality Center, the dialog box that opens is different from the standard file system dialog box. You can switch between the two dialog box versions by clicking the **File System** and **Quality Center** buttons in the Open Shared Object Repository dialog box.

**2** Select the object repository you want to open, and click **Open** or **OK** (depending on whether you are opening it from the file system or a Quality Center project). The object repository opens.

By default, the object repository opens in read-only mode. You can open it in editable format by clearing the **Open in read-only mode** check box in the Open Shared Object Repository dialog box. You can also enable editing for an object repository as described in "Editing Object Repositories" on page 226.

If the object repository is editable, you can add objects to it, modify it, and save it. For more information, see "Managing Objects in Shared Object Repositories" on page 224 and "Saving Object Repositories" on page 221.

---

**Tip:** You can also open an object repository from the **Recent Files** list in the **File** menu.

---

## Saving Object Repositories

After you finish creating or modifying an object repository, you should save it. When you modify an object repository, an asterisk (*) is displayed in the title bar until the object repository is saved.

You can save an object repository to the file system or to a Quality Center project (if you are connected to a Quality Center project). 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 Quality Center project. You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File** > **Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see "Connecting to Your Quality Center Project" on page 44.

All changes you make to an object repository are automatically updated in all 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 components that were open at the time.

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

**To save an object repository:**

1 Make sure that the object repository you want to save is the active window.

2 Choose **File** > **Save** or click the **Save** button. If the file has already been saved, the changes you made are saved. If the file has not yet been saved, the Save Shared Object Repository dialog box opens.

---

**Note:** If you are connected to Quality Center, the dialog box that opens is different from the standard file system dialog box. You can switch between the two dialog box versions by clicking the **File System** and **Quality Center** buttons in the Open Shared Object Repository dialog box.

---

3 Select the folder in which you want to save the object repository.

4 Enter a name for the object repository in the **File name** or **Attachment Name** box (depending on whether you are saving it to the file system or a Quality Center project). Use a descriptive name that will help you easily identify the file.

---

**Note:** You cannot use any of the following characters in the object repository name:
\ / : * " ? < > | ’

---

**5** Click **Save** or **OK** (depending on whether you are saving it to the file system or a Quality Center project).

---

**Notes:**

- When you save a path to a resource, QuickTest checks if the path, or a part of the path, exists in the Folders tab of the Options dialog box (**Tools** > **Options** > **Folders**). 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 tab and define the path relatively.
- For more information, see “Using Relative Paths in QuickTest” on page 324.

---

QuickTest saves the object repository with a **.tsr** extension in the specified location and displays the object repository name and path in the title bar of the repository window.

## Closing Object Repositories

After you finish modifying or using an object repository, you should close it. While an object repository is being edited, it is locked so that it cannot be modified by others. When you close the object repository, it is automatically unlocked. You can also choose to close all open object repositories.

---

**Note:** If you close QuickTest, the Object Repository Manager also closes. If you have made changes that are not yet saved, you are prompted to do so before the Object Repository Manager closes.

---

**To close an object repository:**

1. Make sure that the object repository you want to close is the active window.
2. Choose **File** > **Close** or click the **Close** button in the object repository window's title bar. The object repository is closed and is automatically unlocked. If you have made changes that are not yet saved, you are prompted to do so before the file closes.

**To close all open object repositories:**

Choose **File** > **Close All Windows**, or **Window** > **Close All Windows**. All open object repositories are closed and are automatically unlocked. If you have made changes that are not yet saved, you are prompted to do so before the files close.

## Managing Objects in Shared Object Repositories

You can modify your shared object repositories in a variety of ways to either prepare them for initial use or update them throughout the testing process. You can add and modify objects and object properties in a shared object repository, copy or move objects from one object repository to another, drag objects to a different location in the hierarchy, delete objects, and rename objects. You can also drag and drop test objects from the Object Repository manager to your component. When you modify a shared object repository, an asterisk (*) is displayed in the title bar until the object repository is saved.

**Tip:** You can use the **Edit** > **Undo** and **Edit** > **Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes as necessary. The **Undo** and **Redo** options are related to the active document. When you save an object repository, you cannot undo and redo operations that were performed on that file before the save operation.

If you opened the object repository in read-only mode, you must enable editing for the object repository before you can modify it. This locks the object repository and prevents it from being modified simultaneously by multiple users.

---

**Note:** All changes you make to an object repository are automatically updated in all 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 components that were open at the time. When you open a component on the same computer on which you modified the object repository, the component is automatically updated with all saved changes made in the associated object repository. To see saved changes in a component or repository open on a different computer, you must open the component or object repository file or lock it for editing on your computer to load the changes.

---

---

**Tip:** You can also modify a shared object repository by merging it with another shared object repository. If you merge two shared object repositories, a new shared object repository is created, containing the content of both object repositories. If you merge a shared object repository with a local object repository, the shared object repository is updated with the content of the local object repository. For more information, see Chapter 7, “Merging Shared Object Repositories.”

---

After making sure that your shared object repository is editable, and that it is the active window, you can modify it in the same way as you modify a local object repository. In addition to adding objects to a shared object repository in the same manner as to a local repository, you can also add objects to a shared object repository using the **Navigate and Learn** option. For more information, see:

- "Editing Object Repositories" on page 226
- "Adding Test Objects to Your Component Using the Object Repository Manager" on page 227
- "Adding Test Objects to an Object Repository" on page 156
- "Adding Test Objects Using the Navigate and Learn Option" on page 228
- "Copying, Pasting, and Moving Objects in the Object Repository" on page 166
- "Deleting Objects from the Object Repository" on page 169

## Editing Object Repositories

When you open an object repository, it is opened in read-only mode by default. You can open it in editable format by clearing the **Open in read-only mode** check box in the Open Shared Object Repository dialog box when you open it.

If you opened the object repository in read-only mode, you must enable editing for the object repository before you can modify it. You do not need to enable editing for an object repository if you only want to view it or copy objects from it to another object repository.

When you enable editing for an object repository, it locks the object repository so that it cannot be modified by other users. To enable other users to modify the object repository, you must first unlock it (by disabling edit mode, or by closing it). If an object repository is already locked by another user, if it is saved in read-only format, or if you do not have the permissions required to open it, you cannot enable editing for it.

---

**Note for users of previous QuickTest versions:** If you want to edit an object repository that was created using a version of QuickTest earlier than version 9.0, QuickTest must convert it to the current format before you can edit it. If you do not want to convert it, you can view it in read-only format. After the file is converted and saved, you cannot use it with earlier versions of QuickTest.

---

**To enable editing for an object repository:**

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

2 Choose **File** > **Enable Editing** or click the **Enable Editing** button. The object repository becomes editable.

## Adding Test Objects to Your Component Using the Object Repository Manager

You can drag and drop test objects from the Object Repository Manager to your component. When you drag and drop a test object to your component, QuickTest inserts a step with the default operation for that test object in your component. You cannot drag and drop checkpoint or output objects from the Object Repository Manager.

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

You can also drag and drop test objects from other locations. For more information, see:

➤ "Understanding the Available Keywords Pane" on page 755

➤ "Understanding the Object Repository Window" on page 120

## Adding Test Objects Using the Navigate and Learn Option

The **Navigate and Learn** option enables you to add multiple test objects to a shared object repository while navigating through your application.

Each time you select a window to learn, the selected window and its descendant objects are added to the active shared object repository according to a predefined object filter. You can change the object filter definitions at any time to meet your requirements. The object filter is used for both the **Navigate and Learn** option and the **Add Objects** option. The settings you define are used in both places when you learn objects. For more information on modifying the filter definitions, see "Understanding the Define Object Filter Dialog Box" on page 161.

---

**Note:** The Navigate and Learn option is not supported for environments with mixed hierarchies (object hierarchies that include objects from different environments), for example, `Browser("Homepage").Page("Welcome").AcxButton("Save")` or `Dialog("Edit").AcxEdit("MyEdit")`. To add objects within mixed hierarchies, use other options, as described in "Adding Test Objects to an Object Repository" on page 156.

---

You can use the following keyboard shortcuts when learning objects using the **Navigate and Learn** option:

- **Learn Focused Window.** ENTER
- **Define Object Filter.** CTRL+F
- **Help.** F1
- **Return to Object Repository Manager.** ESC

---

**Note:** Minimized windows are not learned when using the **Navigate and Learn** option.

---

**To add test objects using the Navigate and Learn option:**

1. In the Object Repository Manager, make sure that the object repository to which you want to add objects is the active window and that it is editable.
2. Choose **Object** > **Navigate and Learn** or press F6. The **Navigate and Learn** toolbar opens.


---

**Note:** If this is the first time you are adding objects to the object repository, you may want to change the filter definitions before you continue. You can view the current filter definitions in the **Define Object Filter** button tooltip (displayed in parentheses after the button name). You can change the filter definitions at any time by clicking the **Define Object Filter** button or pressing CTRL+F. For more information, see "Understanding the Define Object Filter Dialog Box" on page 161.

---

3. Click the parent object (for example, Browser, Dialog, Window) you want to add to the object repository to focus it. The **Learn** button on the toolbar is enabled.
4. Click the **Learn** button or focus the **Navigate and Learn** toolbar and press ENTER. A flashing highlight surrounds the focused window and the object and its descendants are added to the object repository according to the defined filter.
5. Navigate in your application to the next window you want to add and then repeat step 4.
6. When you finish adding the required objects to the object repository, click the **Close** button in the **Navigate and Learn** toolbar or press ESC. The **Navigate and Learn** toolbar closes and the Object Repository Manager is redisplayed, showing the objects you just added to the shared object repository.

## Working with Repository Parameters

Repository parameters enable you to specify that certain property values should be parameterized, but leave the actual parameterization to be defined in each component that is associated with the object repository that contains the parameterized test object property values.

Repository parameters are useful when you want to create and run 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.

For example, you may have a button whose text property value changes in a localized application depending on the language of the user interface. You can parameterize the name property value using a repository parameter, and then in each component that uses the object repository you can specify the location from which the property value should be taken. For example, in one component that uses this object repository you can specify that the property value comes from a component parameter, in another component it can come from a local parameter, and in a third component you can specify it as a constant value.

You define all the repository parameters for a specific object repository using the Manage Repository Parameters dialog box. You define each repository parameter together with an optional default value and meaningful description. For more information, see “Managing Repository Parameters” on page 231.

When you open a component that uses an object repository with a repository parameter that has no default value, an indication that there is a repository parameter that needs mapping is displayed in the Missing Resources pane. You can then map the repository parameter as needed in the component. You can also map repository parameters that have default values, and change mappings for repository parameters that are already mapped. For more information on mapping repository parameters, see “Handling Unmapped Shared Object Repository Parameter Values” on page 753.

## Managing Repository Parameters

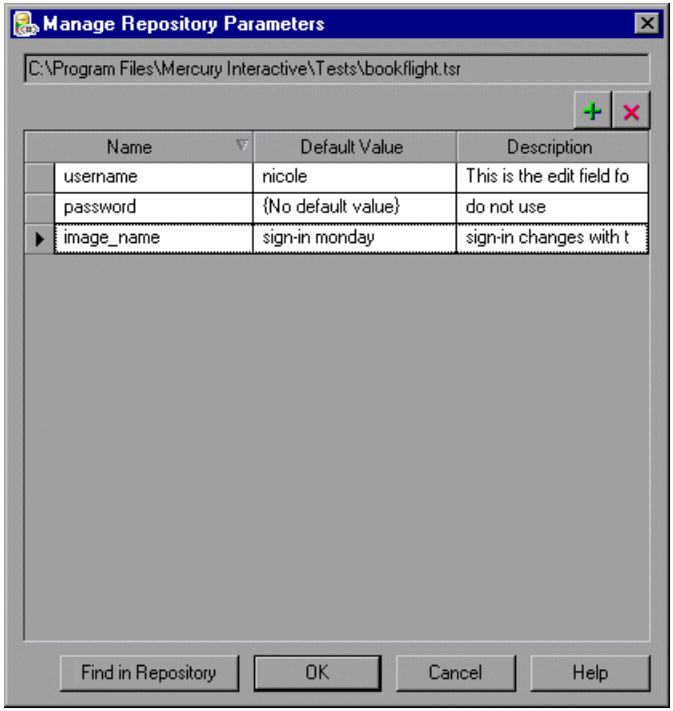
The Manage Repository Parameters dialog box enables you to add, edit, and delete repository parameters for a single shared object repository.

**To manage repository parameters:**

**1** Make sure that the object repository whose parameters you want to manage is the active window.

**2** If the object repository is in read-only format, choose **File** > **Enable Editing** or click the **Enable Editing** button. The object repository becomes editable.

**3** Choose **Tools** > **Manage Repository Parameters** or click the **Manage Repository Parameters** button. The Manage Repository Parameters dialog box opens.



The Manage Repository Parameters dialog box contains the following information and options:

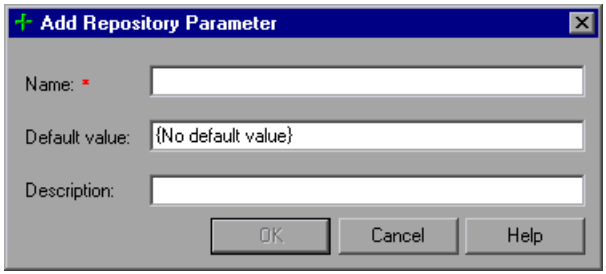
| Option | Description |
|---|---|
| **Repository name** | Displays the name and path of the object repository whose repository parameters you are managing. |
|  | Enables you to add a new repository parameter. For more information, see "Adding Repository Parameters" on page 233. |
|  | Enables you to delete the currently selected repository parameters. For more information, see "Deleting Repository Parameters" on page 235. |
| **Parameter list (Name, Default Value, and Description)** | Displays the list of repository parameters currently defined in this object repository. You can modify a parameter's default value and description directly in the parameter list. For more information, see "Modifying Repository Parameters" on page 234. |
| **Find in Repository** button | Searches for and highlights the first test object in the object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so forth. |

## Adding Repository Parameters

The Add Repository Parameter dialog box enables you to define a new repository parameter. You can also specify a default value for the parameter, and a meaningful description to help identify it when it is used in a component step.

**To add a repository parameter:**

**1** In the Manage Repository Parameters dialog box, click the **Add Repository Parameter** button. The Add Repository Parameter dialog box opens.




**2** In the **Name** box, specify a meaningful name for the parameter. Parameter names must start with an English (Roman) letter and can contain only English (Roman) letters, numbers, and underscores.

**3** In the **Default value** box, you can specify a default value to be used for the repository parameter. This value is used if you do not map the repository parameter to a value or parameter type in a component that uses this object repository. If you do not specify a default value, the repository parameter will appear as unmapped in any components that use this shared object repository.

---

**Tip:** If you specify a default value, you can later remove it by clicking in the **Default Value** cell of the relevant parameter in the Manage Repository Parameters dialog box and then clicking the **Clear Default Value** button. The text **{No Default Value}** is displayed in the cell.

---

4 In the **Description** box, you can enter a description of the repository parameter. The description will help you identify the parameter when mapping repository parameters within a component.

5 Click **OK** to add the parameter to the list of parameters in the Manage Repository Parameters dialog box.

## Modifying Repository Parameters

You can modify the default value of a repository parameter or modify a repository parameter description directly in the Manage Repository Parameters dialog box. However, you cannot modify a repository parameter name.

**To modify a repository parameter:**

1 In the Manage Repository Parameters dialog box, select the required parameter.

2 To modify the default value, click in the **Default Value** cell of the required parameter. You can either modify the default value by entering a new value, or you can remove the default value by clicking the **Clear Default Value** button. If you remove the default value, the text **{No Default Value}** is displayed in the cell. If you do not specify a default value, the repository parameter will appear as unmapped in any components that use this shared object repository.

---

**Note:** If you delete the text manually, it does not remove the default value. It creates a default value of an empty string. You must click the **Clear Default Value** button if you want to remove the default value.

---

3 To modify the parameter description, click in the **Description** cell of the required parameter and enter the required description.

### Deleting Repository Parameters

You can delete a repository parameter definition if it is no longer needed. When you delete a repository parameter that is used in a test object definition, the test object 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 components that have steps using these test objects will fail when you run them.

---

**Tip:** You can use the **Find in Repository** button in the Manage Repository Parameters dialog box to see where a repository parameter is being used.

---

**To delete a repository parameter:**

1 In the Manage Repository Parameters dialog box, select the repository parameters that you want to delete by clicking in the selection area to the left of the parameter name.

2 Click the **Delete Repository Parameter** button. The selected repository parameter is deleted.

## Modifying Object Details

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

After making sure that your shared object repository is editable, and that it is the active window, you modify object details for objects in a shared object repository in the same way as you modify them for local objects. For more information, see:

- "Adding Properties to a Test Object Description" on page 143
- "Defining New Test Object Properties" on page 147
- "Updating Test Object Properties from an Object in Your Application" on page 138
- "Restoring Default Properties for a Test Object" on page 140
- "Removing Properties from a Test Object Description" on page 149
- "Specifying Ordinal Identifiers" on page 150
- "Renaming Test Objects" on page 140

---

**Note:** You can use the **Edit** > **Undo** and **Edit** > **Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes as necessary. The **Undo** and **Redo** options are related to the active document. When you save a repository, you cannot undo and redo operations that were performed on that file before the save operation.

---

You use the Object Repository Manager to specify property values for object descriptions in a shared object repository. The options available when specifying property values for objects in shared object repositories are different from those available when specifying properties for objects in local repositories. For more information on specifying property values for objects in shared object repositories, see "Specifying a Property Value" on page 237.

## Specifying a Property Value

You can specify or modify values for properties in the test object description. You can specify a value using a constant value (either a simple value or a constant value that includes regular expressions) or you can parameterize it using a repository parameter. For more information on repository parameters, see "Working with Repository Parameters" on page 230.

You can also specify or modify values for properties of a checkpoint or output object.

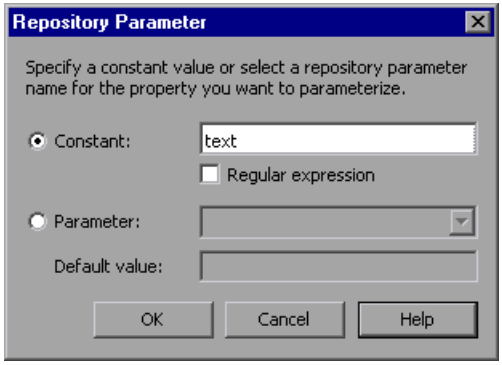
### Specifying and Modifying Values for Properties of a Test Object

You specify or modify the values for properties of a test object in the **Test object details** area.

**To specify a property value of a test object:**

1. Select the test object whose property value you want to specify.
2. In the **Test object details** area, click in the **Value** cell for the required property.
3. Specify the property value in one of the following ways:
   - ➤ If you want to specify a simple constant value, enter it in the **Value** cell. The remaining steps in this procedure are not necessary if you specify a constant value in the **Value** cell. You can also specify a constant value using a regular expression in the Repository Parameter dialog box, as described below.

➤ If you want to parameterize the value using a repository parameter, click the parameterization button in the **Value** cell. The Repository Parameter dialog box opens.




**4** Choose one of the following options to specify a value for the property:

➤ Select the **Constant** radio button and specify a constant value. You can also enter a constant value directly in the **Value** cell of the **Test object details** area. If you used a regular expression in the constant value, select the **Regular expression** check box.

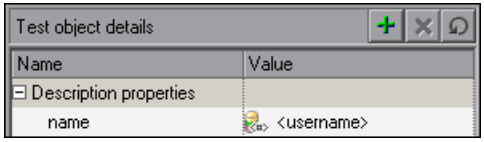
➤ Select the **Parameter** radio button and select a repository parameter from the list of defined parameters. If a default value is defined for the parameter, it is also shown.

---

**Note:** You define repository parameters using the Manage Repository Parameters dialog box. For more information, see “Managing Repository Parameters” on page 231.

---

**5** Click **OK** to close the Repository Parameter dialog box. If you parameterized the value, the parameter name is shown with an icon in the **Value** column of the **Test object details** area, as shown below. Otherwise, the constant value you specified is shown in the **Value** column.




## Specifying and Modifying Values for Properties of a Checkpoint Object

You specify or modify the values for properties of a checkpoint object in the **Object Properties** pane.

**To specify or modify the values for properties of a checkpoint object:**

**1** Select the checkpoint object whose property values you want to specify or modify from the **Checkpoint and Output Objects** tree.

**2** Specify or modify the values for properties of a checkpoint object the same way as you do in the relevant checkpoint properties dialog box.

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

- "Understanding the Checkpoint Properties Dialog Box" on page 559
- "Checking a Bitmap" on page 566

### Specifying and Modifying Values for Properties of an Output Object

You specify or modify the values for properties of an output object in the **Object Properties** pane.

**To specify or modify the values for properties of an output object:**

1. Select the output object whose property values you want to specify or modify from the **Checkpoint and Output Objects** tree.
2. Specify or modify the values for properties of an output object the same way as you do in the relevant output value properties dialog box.

   For more information on specifying and modifying values for properties of an output object, see "Defining Standard Output Values" on page 575.

## Locating Test Objects

You can search for a specific test object in your object repository in several ways. You can search for a test object according to its type. For example, you can search for a specific edit box, or you can point to an object in your application to automatically highlight that same object in your repository. You can replace specific property values with other property values. For example, you can replace a property value userName with the value user name. You can also select an object in your object repository and highlight it in your application to check which object it is.

After making sure that your shared object repository is the active window, you locate an object in a shared object repository in the same way as you locate it in a local object repository. If you want to replace property values, you must also make sure that the object repository is editable.

For more information, see:

- "Finding Objects in an Object Repository" on page 170
- "Highlighting an Object in Your Application" on page 173
- "Locating a Test Object in the Object Repository" on page 174

# Performing Merge Operations

The Object Repository Merge Tool enables you to merge test objects from the local object repository of one or more 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 test objects locally in a specific component and want to add them to the shared object repository so they are available to all components that use that object repository. You can also use the Object Repository Merge Tool to merge two shared object repositories into a single shared object repository.

You open the Object Repository Merge Tool by choosing **Tools** > **Object Repository Merge Tool** in the Object Repository Manager. For more information on performing merge operations and updating object repositories with local objects, see Chapter 7, "Merging Shared Object Repositories."

---

**Notes:**

- While the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager.
- The Object Repository Merge Tool does not merge checkpoint and output objects.

---

# Performing Import and Export Operations

You can import and export object repositories from and to XML files. XML provides a structured, accessible format that enables you to make changes to object repositories using the XML editor of your choice and then import them back into QuickTest. You can view the required format for the object repository in the QuickTest Object Repository Schema Help (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Object Repository Schema**), or by exporting a saved object repository.

You can import and export files either from and to the file system or a Quality Center project (if QuickTest is connected to Quality Center).

You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File > Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see "Connecting to Your Quality Center Project" on page 44.

## Importing from XML

You can import an XML file (created using the required format) as an object repository. For information on the XML format, see "Understanding the XML File Structure" on page 244. The XML file can either be an object repository that you exported to XML format using the Object Repository Manager, or an XML file created using a tool such as QuickTest Siebel Test Express or a custom built utility. You must adhere to the XML structure and format.

---

**Tip:** To view the required XML structure and format, see the *QuickTest Object Repository Schema Help* (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Object Repository Schema**). You can also export an existing shared object repository to XML and then use the XML file as a guide. For more information, see "Exporting to XML" on page 243.

---

**To import from XML:**

**1** Choose **File** > **Import from XML**. The Import from XML dialog box opens.

---

**Notes:**

- If you are connected to Quality Center, the dialog box that opens is different from the standard file system dialog box. You can switch between the two dialog box versions by clicking the **File System** and **Quality Center** buttons in the Import from XML dialog box.
- Checkpoint and output objects are not included when importing the contents of an object repository from an XML file.

---

**2** Select the XML file you want to import, and click **Open** or **OK** (depending on whether you are opening it from the file system or a Quality Center project).

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

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

## Exporting to XML

You can export the test objects in an object repository to an XML file. This enables you to edit it using any XML editor, and also enables you to save it in an accessible, versatile format.

**To export to XML:**

**1** Make sure that the object repository whose test objects you want to export is the active window.

**2** Make sure that the object repository is saved.

**3** Choose **File** > **Export Test Objects to XML**. The Export to XML dialog box opens.

---

**Notes:**

- If you are connected to Quality Center, the dialog box that opens is different from the standard file system dialog box. You can switch between the two dialog box versions by clicking the **File System** and **Quality Center** buttons in the Export to XML dialog box.
- Checkpoint and output objects are not included when exporting the contents of an object repository to an XML file.

---

**4** Select the location in which to save the file, specify the file or attachment name, and click **Save** or **OK** (depending on whether you are saving it to the file system or a Quality Center project).

**5** The test objects of the object repository are exported to the specified XML file and a summary message box opens showing information regarding the number of objects, parameters, and metadata that were successfully exported to the specified file.

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

## Understanding the XML File Structure

QuickTest uses a defined XML schema for object repositories. You must follow this schema when creating or modifying object repository files in XML format. The schema of this file is documented in the *QuickTest Object Repository Schema Help* (**Help** > **QuickTest Professional Help** > **QuickTest Advanced References** > **QuickTest Object Repository Schema**).

# Managing Object Repositories Using Automation

QuickTest provides an Object Repository automation object model that enables you to manage QuickTest shared object repositories and their contents from outside of QuickTest. The automation object model enables you to use a scripting tool to access QuickTest shared object repositories via automation.

Just as you use the QuickTest Professional automation object model to automate your QuickTest operations, you can use the objects and methods of the Object Repository automation object model to write scripts that manage shared object repositories, instead of performing these operations manually using the Object Repository Manager. For example, you can add, remove, and rename test objects; import from and export to XML; retrieve and copy test objects; and so forth.

After you have retrieved a test object, you can manipulate it using the methods and properties available for that test object class. For example, you can use the GetTOProperty and SetTOProperty methods to retrieve and modify its properties. For more information on available test object methods and properties, see the *HP QuickTest Professional Object Model Reference*.

Automation programs are especially useful for performing the same tasks multiple times or on multiple object repositories. You can write your automation scripts in any language and development environment that supports automation. For example, you can use VBScript, JavaScript, Visual Basic, Visual C++, or Visual Studio.NET. For general information on controlling QuickTest using automation, see "Automating QuickTest Operations" on page 817.

## Using the QuickTest Professional Object Repository Automation Reference

The QuickTest Professional Object Repository Automation Reference is a Help file that provides detailed descriptions, syntax information, and examples for the objects and methods in the QuickTest object repository automation object model.

The Help topic for each automation object includes a list and description of the methods associated with that object. Method Help topics include detailed description, syntax, return value type, and argument value information.

You can open the *QuickTest Professional Object Repository Automation Reference* from the main QuickTest Help (**Help** > **QuickTest Professional Help** > **QuickTest Advanced References** > **QuickTest Object Repository Automation**).

---

**Note:** The syntax and examples in the Help file are written in VBScript-style. If you are writing your automation program in another language, the syntax for some methods may differ slightly from what you find in the corresponding Help topic. For information on syntax for the language you are using, see the documentation included with your development environment or to general documentation for the programming language.

---

# 7

# Merging Shared Object Repositories

QuickTest Professional enables you to merge two shared object repositories into a single shared object repository using the Object Repository Merge Tool. You can also use this tool to merge objects from the local object repository of one or more actions into a shared object repository.

**This chapter includes:**

# About Merging Shared Object Repositories

When you have multiple shared object repositories that contain test objects from the same area of your application, it may be useful to combine those test objects into a single object repository for easier maintenance. You could do this by moving or copying objects in the Object Repository Manager. However, if you have test objects in different object repositories that represent the same object in your application, and the descriptions for these objects in the different object repositories are not identical, it may be difficult to recognize and handle these conflicts.

The Object Repository Merge Tool helps you to solve the above problem by merging two selected object repositories for you and providing options for addressing test objects with conflicting descriptions. Using this tool, you merge two shared object repositories (called the **primary** object repository and the **secondary** object repository), into a new third object repository, called the **target** object repository. Objects in the primary and secondary object repositories are automatically compared and then added to the target object repository according to preconfigurable rules that define the defaults for how conflicts between objects are resolved.

After the merge process, the Object Repository Merge Tool provides a graphic presentation of the original objects in the primary and secondary object repositories, which remain unchanged, as well as the objects in the merged target object repository. Objects that had conflicts are highlighted. The conflict of each object that you select in the target object repository is described in detail. The Object Repository Merge Tool provides specific options that enable you to keep the default resolution for each conflict, or modify conflict resolutions individually, according to your requirements.

The Object Repository Merge Tool also enables you to merge objects from the local object repository of one or more actions into a shared object repository. For example, if QuickTest learned objects locally in a specific action in your test, you may want to add the objects to the shared object repository, so that they are available to all actions in different tests that use that object repository.

---

**Notes:**

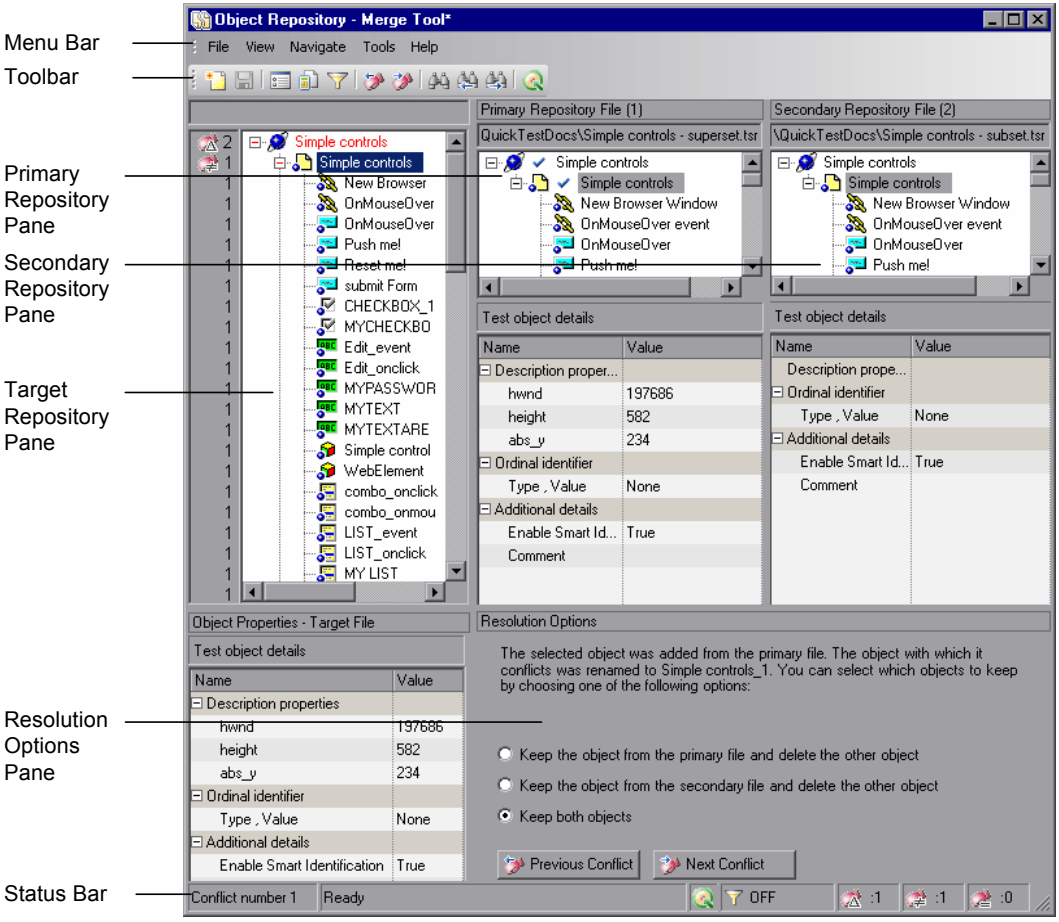
- The Object Repository Merge Tool does not merge checkpoint or output objects from the primary and secondary object repositories into the target shared object repository. You can copy or manually move these objects to your target object repository after you complete the merge process, using the Object Repository Manager.
- When the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager or Object Repository Comparison Tool. For more information on the Object Repository Manager, see Chapter 6, "Managing Object Repositories."

---

# Understanding the Object Repository Merge Tool

You open the Object Repository Merge Tool by choosing **Tools** > **Object Repository Merge Tool** in the Object Repository Manager.

An example of the Object Repository - Merge Tool window is shown below:



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

- **Menu bar.** Displays menus of Object Repository Merge Tool commands. These commands are described in various places throughout this chapter. Shortcut keys for menu commands are described in "File Menu Commands" on page 257.
- **Toolbar.** Contains buttons of commonly used menu commands to assist you in merging, managing, and saving object repositories. Toolbar buttons are described in "Using Toolbar Commands" on page 256.
- **Target Repository Pane.** Displays the objects that were merged from the primary and secondary object repositories. You can also choose to show or hide the Target Repository Object Properties pane, which displays the properties of any object that is selected in the Target Repository pane. For more information, see "Target Repository Pane" on page 252.
- **Primary Repository Pane.** Displays the objects in the primary object repository. For more information, see "Primary and Secondary Repository Panes" on page 254.
- **Secondary Repository Pane.** Displays the objects in the secondary object repository. For more information, see "Primary and Secondary Repository Panes" on page 254.
- **Resolution Options Pane.** Provides source, conflict, and resolution details about the objects in the target object repository pane, and enables you to modify how a selected conflict is resolved. For more information, see "Resolution Options Pane" on page 254.
- **Status Bar.** Provides source, conflict, and resolution details about the object selected in the target object repository pane, the filter status, and an icon legend. For more information, see "Status Bar" on page 255.

## Changing the View

You can change the view presented by the Object Repository Merge Tool according to your working preferences.

- Drag the edges of the panes to resize them in the Object Repository Merge Tool window.
- Choose **Primary Repository**, **Secondary Repository**, **Target Repository Object Properties**, or **Resolution Options** from the **View** menu to hide or show these panes in the Object Repository Merge Tool.
- Choose **View** > **Set as Default Layout** to set your current view as the default view, which displays each time you open the Object Repository Merge Tool. You can choose **View** > **Restore Default Layout** to restore the view to the default settings after you make changes.

## Target Repository Pane

The target object repository pane displays a hierarchy of the objects, as well as their respective properties and values, that were merged from the primary and secondary object repositories. In the column to the left of the object hierarchy, the pane displays the source file of each object (**1** is displayed for the primary file and **2** for the secondary file), and an icon representing the type of conflict, if any.

When you save the target object repository, the file path is displayed above the object hierarchy.

---

**Note:** To make it easier to see the status of an object at a glance, the text colors of the object names in the target object repository can be set according to their source and whether they caused a conflict. For more information, see “Specifying Color Settings” on page 264.

---

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 choose an option to expand or collapse the entire hierarchy in the target object repository, or, when applicable, to change the conflict resolution method and result.
- You can expand or collapse the hierarchy of the node by double-clicking a node. You can also expand or collapse the entire hierarchy in the target object repository by choosing **Collapse All** or **Expand All** from the **View** menu.
-  You can jump directly to the next or previous conflict in the target object repository hierarchy by choosing **Next Conflict** or **Previous Conflict** from the **Navigate** menu, or by clicking the **Next Conflict** or **Previous Conflict** buttons in the toolbar or Resolution Options pane.
- You can locate one or more objects in the target object repository by using the Find dialog box. For more information, see “Finding Specific Objects” on page 283.
- 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 action's local object repository when the merge is complete.
- You can show or hide the pane by selecting or clearing **Resolution Options** in the **View** menu.

### Status Bar

The status bar shows the following information about the merge process and the results that are displayed:

- The conflict number (if any) of the object selected in the target object repository pane.
- A progress bar, displayed during the merge process. **Ready** is displayed when the process is complete.

- The the Quality Center icon, displayed when QuickTest is connected to a Quality Center project.

- The filter status, shown next to the Filter icon: **OFF** indicates that the object repositories are not filtered and all objects are shown. **ON** indicates a filter is active and that some objects may have been filtered out of the display.
- A legend of the icons used in the target object repository pane. The following icons may be displayed:

  - Similar Description Conflict

  - Same Name Different Description Conflict

  - Same Description Different Name Conflict

For more information on conflict types, see "Understanding Object Conflicts" on page 276.

**Tips**:

- Position your pointer over a conflict icon in the status bar to see a tooltip description of the conflict type.
- Click any of the conflict icons to view the Statistics dialog box. For more information, see "Viewing Merge Statistics" on page 275.
- Click the **Filter** icon in the status bar to view the Filter dialog box. The filter is shown as **ON** in the status bar when a filter is currently in use. For more information, see "Filtering the Target Repository Pane" on page 281.


# Using Object Repository Merge Tool Commands

You can select Object Repository Merge Tool commands from the menu bar or from the toolbar. You can perform certain commands by pressing shortcut keys. You can also select an object in the target object repository pane and choose commands from the context-sensitive (right-click) menu.

## Using Toolbar Commands

You can perform frequently used commands by clicking buttons in the Object Repository Merge Tool toolbar.

| | Description |
|---|---|
|  | New Merge (described in "File Menu Commands" on page 257) |
|  | Save (described in "File Menu Commands" on page 257) |
|  | Settings (described in "Tools Menu Commands" on page 260) |
|  | Statistics (described in "View Menu Commands" on page 258) |
|  | Filter (described in "Tools Menu Commands" on page 260) |

| | Description |
|---|---|
|  | Previous Conflict (described in "Navigate Menu Commands" on page 260) |
|  | Next Conflict (described in "Navigate Menu Commands" on page 260) |
|  | Find (described in "Navigate Menu Commands" on page 260) |
|  | Find Previous (described in "Navigate Menu Commands" on page 260) |
|  | Find Next (described in "Navigate Menu Commands" on page 260) |
|  | Quality Center Connection (described in "File Menu Commands" on page 257) |

## Performing Object Repository Merge Tool Commands

You can perform frequently-used commands by clicking toolbar buttons or choosing the relevant menu option. You can also perform some commands by pressing the relevant shortcut keys.

### File Menu Commands

You can manage your merged object repository using the following **File** menu commands:

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **New Merge** | CTRL+N | Enables you to specify two object repositories with which to perform a new merge operation. |
|  | **Save** | CTRL+S | Saves the merged shared object repository. |
| | **Save As** | | Opens the Save Shared Object Repository dialog box, enabling you to specify a name, file type, and storage location for the merged shared object repository. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Quality Center Connection** | | Enables you to connect QuickTest to a Quality Center project. For more information, see "Connecting to Your Quality Center Project" on page 44. |
| | **Exit** | | Closes the Object Repository - Merge Tool window. (Also prompts you to save the merged object repository if you did not yet save it.) |

## View Menu Commands

You can manage the way that the Object Repository Merge Tool is displayed on your screen using the following **View** menu commands:

| | Command | Function |
|---|---|---|
| | **Primary Repository** | Displays the Primary Repository File pane, containing a hierarchical view of the objects from the first source object repository that you chose to merge. Also displays the details for each object selected in this pane. For more information, see "Primary and Secondary Repository Panes" on page 254 and "Merging Two Object Repositories" on page 266. |
| | **Secondary Repository** | Displays the Secondary Repository File pane, containing a hierarchical view of the objects from the second source object repository that you chose to merge. Also displays the details for each object selected in this pane. For more information, see "Primary and Secondary Repository Panes" on page 254 and "Merging Two Object Repositories" on page 266. |
| | **Target Repository Object Properties** | Displays the Object Properties - Target File pane, which displays the details for each test object selected in the target repository pane. For more information, see "Target Repository Pane" on page 252. |

| | Command | Function |
|---|---|---|
| | **Resolution Options** | Displays the Resolution Options pane, which provides information about any conflict that occurred during the merge. For more information, see "Resolution Options Pane" on page 254 and "Resolving Object Conflicts" on page 279. |
| | **Restore Default Layout** | Restores the view that you saved using the **Set as Default Layout** option (described below). This is useful if you resize a pane, or show or hide specific panes and then want to restore your saved view. For more information, see "Changing the View" on page 252. |
| | **Set as Default Layout** | Enables you to save the current view so that each time you open the Object Repository - Merge Too, this view is displayed. If you later modify this view by resizing panes, or showing or hiding them, you can restore your default view using the **Restore Default Layout** option (described above). For more information, see "Changing the View" on page 252. |
|  | **Statistics** | Opens the Statistics dialog box, which describes how the files were merged, and the number and type of any conflicts that were resolved during the merge. For more information, see "Viewing Merge Statistics" on page 275. |
| | **Collapse All** | Collapses the entire hierarchy in the Target Object Repository pane.<br>**Tip:** You can collapse a single node by double-clicking it. |
| | **Expand All** | Expands the entire hierarchy in the Target Object Repository pane.<br>**Tip:** You can expand a single node by double-clicking it. |

## Navigate Menu Commands

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Next Conflict** | F4 | Finds the next conflicting object in the merged object repository. |
|  | **Previous Conflict** | SHIFT+F4 | Finds the previous conflicting object in the merged object repository. |
|  | **Find** | CTRL+F | Opens the Find dialog box. |
|  | **Find Next** | F3 | Finds the next object in the merged object repository according to the search specifications in the Find dialog box. |
|  | **Find Previous** | SHIFT+F3 | Finds the previous object in the merged object repository according to the search specifications in the Find dialog box. |

## Tools Menu Commands

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

| | Command | Function |
|---|---|---|
|  | **Settings** | Opens the Settings dialog box, enabling you to:<br>➤ Configure how the Object Repository Merge Tool deals with conflicting objects during a merge<br>➤ Specify the text color of the object names displayed in the target object repository<br>For more information, see "Defining Default Settings" on page 261. |
|  | **Filter** | Opens the Filter dialog box, enabling you to show all of the test objects in the Target Repository pane, or show only the objects that had conflicts that were resolved during the merge. For more information, see "Filtering the Target Repository Pane" on page 281. |

### Help Menu Command

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

| Command | Shortcut Key | Function |
|---|---|---|
| **Object Repository Merge Tool Help** | F1 | Opens the Object Repository Merge Tool Help. |

## Defining Default Settings

The Object Repository Merge Tool is supplied with predefined settings that are used when merging object repositories or when updating a shared object repository from local object repositories. These are the default settings:

- Configure how the Object Repository Merge Tool deals with conflicting objects in the primary and secondary object repositories (or local and shared object repositories when updating a shared object repository from local object repositories).
- Specify the text color of the object names that are displayed in the target object repository.

You can change these settings at any time to create new default settings. After you change the settings, all new merges are performed according to the new default settings.

---

**Tip:** If you want to change the settings before merging two object repositories, you must click **Cancel** to close the New Merge dialog box, change the settings as described in the next sections, and then perform the merge.

---

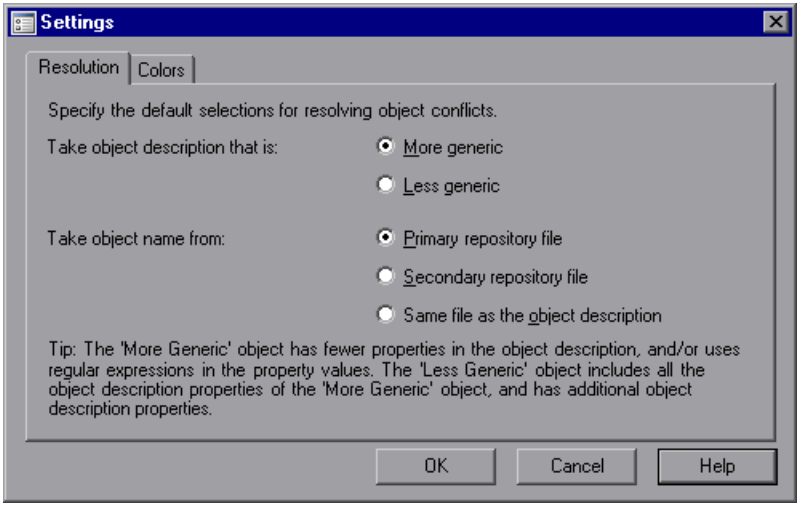
## Specifying Default Resolution Settings

You can configure how the Object Repository Merge Tool automatically deals with conflicting objects during the merge process or when performing an **Update from Local Repository** operation.

**To specify default resolution settings:**

**1** Choose **Tools** > **Settings** or click the **Settings** button. The Settings dialog box opens.

**2** Click the **Resolution** tab.



**3** Select the appropriate radio buttons to specify the default resolution settings that the Object Repository Merge Tool applies when dealing with conflicting objects.

- **Take object description that is.** Specifies how to resolve conflicts in which two objects have the same name, but their descriptions differ. You can specify that the target object repository takes the object description that is more generic or less generic.
  - **More generic.** Instructs the Object Repository Merge Tool to take the object that has fewer identifying properties than the object with which it conflicts, or uses regular expressions in its property values. This is the default setting.
  - **Less generic.** Instructs the Object Repository Merge Tool to take the object that has all the identifying properties of the object with which it conflicts, plus additional identifying properties.
- **Take object name from.** Specifies how to resolve conflicts where two objects have the same or similar descriptions, but their names differ. You can select the source from which the target object repository takes the object name:
  - **Primary repository file.** The target object repository takes the object name from the object in the primary object repository. This is the default setting. (When updating a shared object repository from a local object repository, this option is for the **Local object repository**.)
  - **Secondary repository file.** The target object repository takes the object name from the object in the secondary object repository. (When updating a shared object repository from a local object repository, this option is for the **Shared object repository**.)
  - **Same file as the object description.** The target object repository takes the object name from the object in the same object repository from which it took the object description.

---

**Note:** When updating a shared object repository from a local object repository, the object repositories are referred to as the Local and Shared object repository.

---

**4** Click **OK**. The Object Repository Merge Tool will apply your selections when resolving conflicts between objects in all future object repository merges.

---

**Note:** If you make any change to the resolution settings while a merged object repository is open, you are asked whether you want to merge the open files again with the new settings. Click **Yes** to merge the files again with the new settings, or click **No** to keep the existing merge created with the previous settings. If you click **No**, the new settings will apply only to future merges.

---

## Specifying Color Settings

You can specify the color in which object names are displayed in the target object repository according to their source, and whether they caused a conflict. This enables you to see the status of each object more easily.

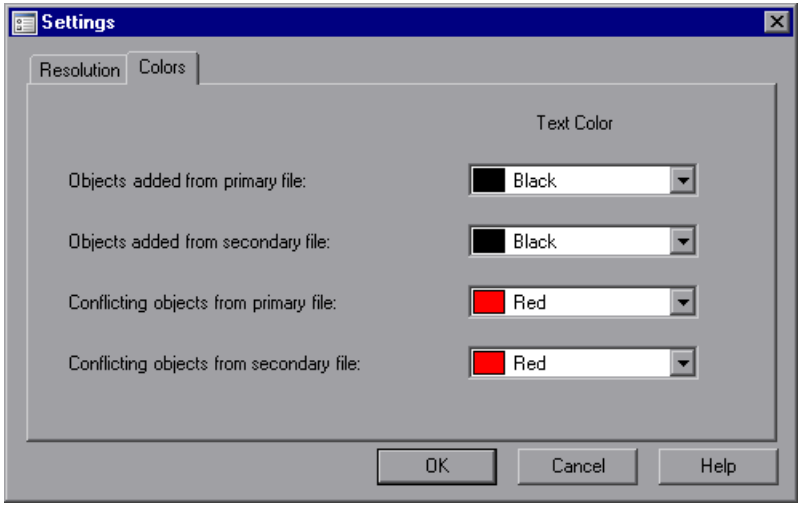
---

**Note:** The options in the Colors tab of the Settings dialog box apply equally to objects added from the local (primary) and shared (secondary) object repositories, when performing an **Update from Local Repository** operation.

---

**To specify color settings:**

**1** Choose **Tools** > **Settings** or click the **Settings** button. The Settings dialog box opens.




**2** For each item in the Colors tab, click the down arrow next to the text box and select an identifying color from the Custom, Web, or System tabs.

**3** Click **OK**. Object names in the target object repository are displayed in the selected color according to your selections.

# Merging Two Object Repositories

Using the Object Repository Merge Tool, you can merge two source object repositories to create a new shared object repository. Objects in the object repositories are automatically compared and added to the new object repository according to configurable rules that define how conflicts between objects are resolved. The original source files are not changed.

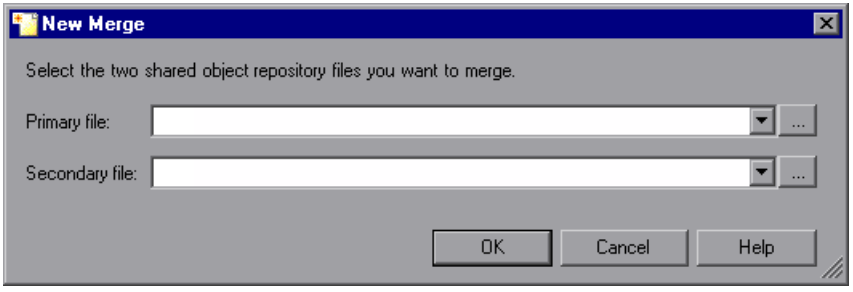
---

**Note:** An object repository that is currently open by another user is locked. If you try to merge the locked file, a warning message displays, but you can still perform the merge because the merge process does not modify the source files. Note that changes made to the locked file by the other user may not be included in the merged object repository.

---

**To merge two object repositories:**

**1** In the Object Repository Manager, choose **Tools** > **Object Repository Merge Tool**. The New Merge dialog box opens on top of the Object Repository - Merge Tool window.



**Tips:**

- If the Object Repository - Merge Tool window is already open, you can choose **File** > **New Merge** or click the **New Merge** button to open the New Merge dialog box.
- If you want to change the configured settings before merging the object repositories, click **Cancel** to close the New Merge dialog box, change the settings as described in "Defining Default Settings" on page 261, and then perform the merge.

**2** In the **Primary file** and **Secondary file** boxes, enter (or browse to) the **.tsr** object repositories that you want to merge into a single object repository. You can click the down arrow next to each box to view and select recently used files.

**Notes:**

- It is recommended that you select as your primary object repository the object repository in which you have invested the most effort, meaning the object repository with more objects, object properties, and values.

- A warning icon is displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. You can position your pointer over the icon to see a tooltip explanation of the error. Enter or select an existing **.tsr** file with the correct path.
- If you want to merge an object repository that was created using a version of QuickTest earlier than version 9.0, you must first open and save it in the Object Repository Manager to update it to the new format.
- If you are connected to Quality Center, you can enter (or browse to) object repositories from Quality Center as well as from the file system.

3 Click **OK**. The Object Repository Merge Tool automatically merges the selected object repositories into a new target object repository according to the configured resolution settings, and displays the results in the Statistics dialog box on top of the Object Repository - Merge Tool window.

4 Review the merge statistics, as described in "Viewing Merge Statistics" on page 275, and click **Close**.

In the Object Repository - Merge Tool window, you can:

- Modify any conflict resolutions between objects from the source object repositories, if necessary, as described in "Resolving Object Conflicts" on page 279.
- Filter the objects in the target object repository, as described in "Filtering the Target Repository Pane" on page 281.
- Save the target object repository to the file system or to a Quality Center project, as described in "Saving the Target Object Repository" on page 284.

## Updating a Shared Object Repository from Local Object Repositories

You can update a shared object repository by merging local object repositories associated with one or more components (via the application area) 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.

In the merge process, the objects in the local object repository for the selected component are moved to the target shared object repository. The component then uses the objects from the updated shared object repository.

You can view or change how conflicting objects are dealt with during the update process in the Settings dialog box. For more information, see "Defining Default Settings" on page 261.

If you choose to add local object repositories for more than one component, QuickTest performs multiple merges, merging each component's local object repository with the target object repository one at a time, for all the components in the list. You can view and modify the results of each merge if necessary.

---

**Notes:**

The Object Repository Merge Tool does not merge checkpoint or output objects from a local object repository into the target shared object repository. You can export checkpoint or output objects from a local object repository to a shared object repository and then manually move the checkpoint and output objects from the exported object repository to your target object repository after you complete the merge process, using the Object Repository Manager.

You can merge local object repositories only from components whose application areas are associated with the shared object repository you are updating.

---

**To update a shared object repository from a local object repository:**

**1** Choose **Resources** > **Object Repository Manager**. The Object Repository Manager opens.

---

**Note:** For more information on the Object Repository Manager, see Chapter 6, "Managing Object Repositories."

---

**2** In the Object Repository Manager, choose **File** > **Open** or click the **Open** button. The Open Shared Object Repository dialog box opens.

If you are currently connected to a Quality Center project, the Open Shared Object Repository dialog box displays the component tree for the project. Select a component to view the shared object repositories attached to the component.

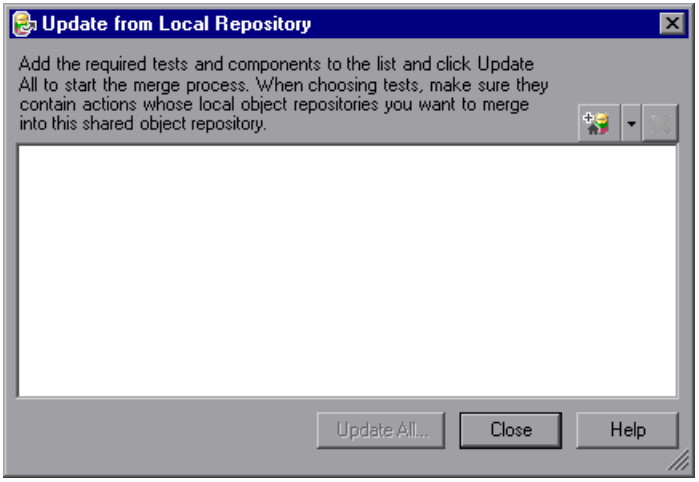
**3** Browse to the **.tsr** file that contains the shared object repository you want to update, clear the **Open in read-only mode** check box, and click **Open**, or click **OK** in the case of Quality Center attached files. The file opens with the objects and properties displayed in editable format.

---

**Tip:** If you opened the object repository in read-only mode, choose **File > Enable Editing** or click the **Enable Editing** button in the Object Repository Manager toolbar. The object repository file is made editable.

---

**4** Choose **Tools > Update from Local Repository**. The Update from Local Repository dialog box opens.



**5** Make sure you are connected to your Quality Center project. Click the down arrow next to the **Add Tests** button, and choose **Browse for Component**. The Open QuickTest Component from Quality Center Project dialog box opens.

Browse to the component whose local object repository you want to merge into the shared object repository.

---

**Note:** You can only add a component whose application area is associated with the shared object repository you are updating and whose local object repository contains objects.

---

**6** Repeat step5 to add additional components if required.

---

**Note:** The local object repositories associated with all the components are included in the merge. If you want to remove an component from the merge, select it in the list and click **Delete**.

---

**7** Click **Update All**. QuickTest automatically merges the first component local object repository into the shared object repository according to the configured settings, and displays the results in the Statistics dialog box on top of the Object Repository Merge Tool window.

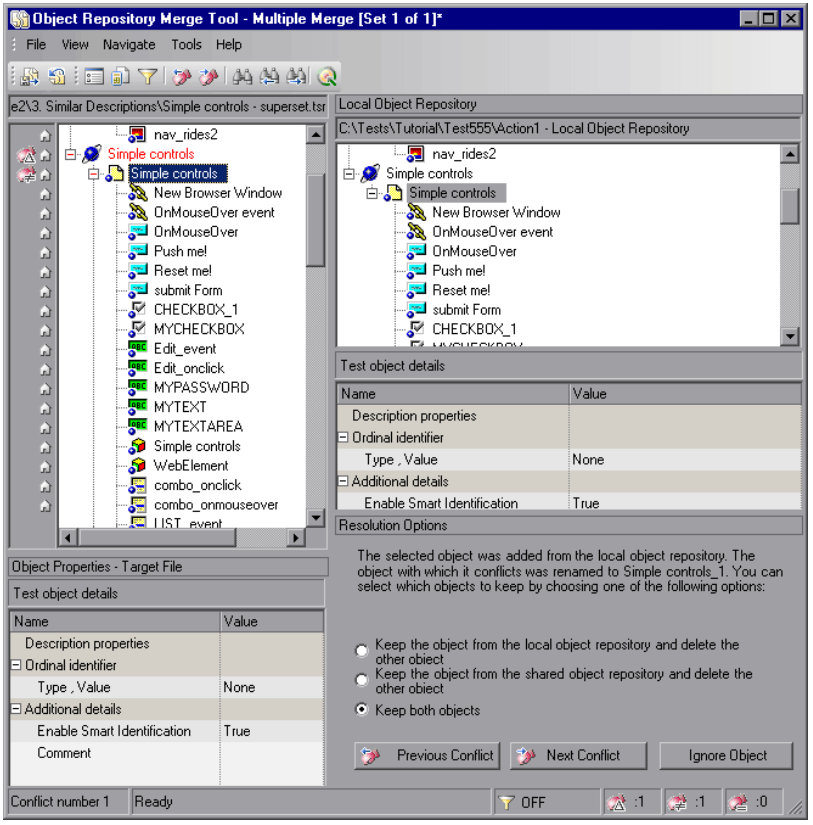
---

**Note:** Before each merge, QuickTest checks whether the local object repository is in use by another user. If so, the local object repository is locked and the objects for the selected 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.

---

**8** Review the merge statistics, as described in “Viewing Merge Statistics” on page 275, and click **Close**.

The Object Repository - Merge Tool window for a local object repository merge displays the local object repository as the primary object repository, and the shared object repository as the target object repository.



At the left of each object in the target object hierarchy is an icon that indicates the source of the objects:

 indicates that the object was added from the local object repository.

 indicates that the object already existed in the shared object repository.

---

**Note:** If you specified more than one component in the Update from Local Repository dialog box, QuickTest performs multiple merges, merging each component's local object repository with the target object repository one at a time. The Statistics dialog box and the Object Repository Merge Tool - Multiple Merge window displayed after this step show the merge results of the first merge (the local object repository of the first component being merged into the shared object repository). QuickTest enables you to view, and modify if necessary, the results of each merge in sequence. The number of each merge set in a multiple merge is displayed in the title bar, for example, [Set 2 of 3].

---

**9** For each object merged into the shared object repository, you can accept the automatic merge or use the Resolution Options pane to:

- ➤ Keep a specific object from the shared object repository and delete the conflicting object from the local object repository.
- ➤ Keep a specific object from the local object repository and delete the conflicting object from the shared object repository.
- ➤ Keep conflicting objects from both the shared object repository and the local object repository.
- ➤ Exclude a specific local repository object from the merge process so that it is not included in the shared object repository. Select the object in the Shared Object Repository pane and click **Ignore Object** at the bottom of the Resolution Options pane. The object is removed from the shared object repository and grayed in the local object repository tree. It remains in the action's local object repository when the merge is complete.

---

**Notes:**

- ➤ The **Ignore Object** button is only visible in the Merge Tool window for a local object repository merge, and is only enabled when an object in the local object repository is selected.
- 
➤ The **Ignore Object** operation cannot be reversed. To include the object again in the merge process, you must repeat the merge by clicking **Revert to Original Merged Files** in the toolbar.

---

For more information, see "Resolving Object Conflicts" on page 279.


**10** If you are performing multiple merges, click the **Save and Merge Next** button in the Object Repository Merge Tool toolbar to perform the next merge (the local object repository of the next component being merged into the shared object repository).

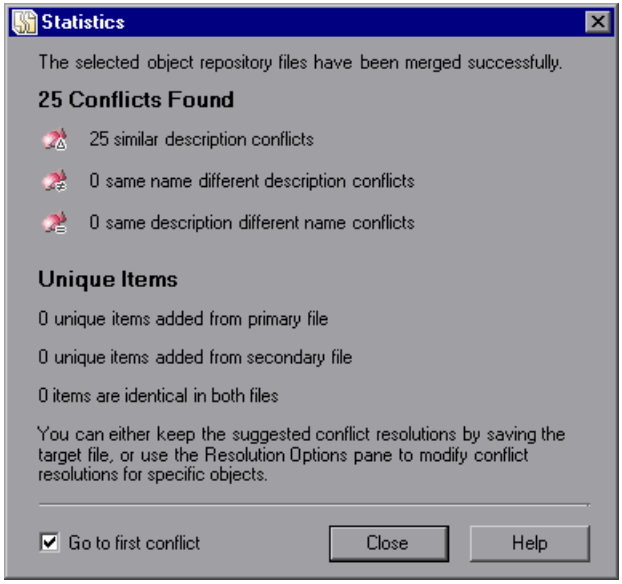
**11** Click **Yes** to save your changes between merges. If you click **No**, the current merge (objects merged from the last component) will not be saved.

**12** Repeat steps 8 through 11 to complete the multiple merges.

**13** Choose **File** > **Exit**, then click **Yes** to save the updated object repository.

# Viewing Merge Statistics

After you merge two object repositories, the Object Repository Merge Tool displays the Statistics dialog box, which describes how the files were merged, and the number and type of any conflicts that were resolved during the merge.




---

**Note:** The Statistics dialog box shown after performing an **Update from Local Repository** merge differs slightly from the dialog box shown above.

---

**Tip:** You can view the merge statistics in the Statistics dialog box at any time by choosing **View > Statistics** in the Object Repository - Merge Tool window, by clicking the **Statistics** button in the toolbar, or by clicking a conflict icon in the status bar.

---

The Statistics dialog box displays the following information:

- The number and type of any conflicts between the objects added to the target object repository. Conflict types are described in "Resolving Object Conflicts" on page 279.
- The number of items added to the target object repository that are unique in each of the primary or secondary (or local) files, or are identical in both files.

---

**Tip:** Select the **Go to first conflict** check box to jump to the first conflict in the target object repository immediately after you close the Statistics dialog box.

---

## Understanding Object Conflicts

Merging two object repositories can result in conflicts arising from similarities between the objects they contain. The Object Repository Merge Tool identifies three possible conflict types:

- **Similar Description Conflict.** Two objects which have the same name and the same object hierarchy, but which have slightly different descriptions. In this conflict type, one of the objects always has a subset of the properties set of the other object. These conflicts are described on page 277.

  By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object that has fewer identifying properties than the object with which it conflicts. For information on changing the default settings, see "Defining Default Settings" on page 261.

- **Same Name Different Description Conflict.** Two objects which have the same name and the same object hierarchy, but differ somehow in their description (for example, they have different properties, or the same property with different values). These conflicts are described on page 278.

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object from both files. The object that is added from the secondary file is renamed by adding an incremental numeric suffix to the name, for example, Edit_1. For information on changing the default settings, see "Defining Default Settings" on page 261.

➤ **Same Description Different Name Conflict.** Two objects which have identical descriptions, have the same object hierarchy, but differ in their object names. These conflicts are described on page 279.

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object name from the primary source file. For information on changing the default settings, see "Defining Default Settings" on page 261.

---

**Note:** Objects that do not have a description, such as Page or Browser objects, are compared by name only. If the same object is contained in both the source object repositories but with different names, they will be merged into the target object repository as two separate objects.

---

### Similar Description Conflict

An object in the primary object repository and an object in the secondary object repository have the same name, and they have similar, but not identical, description properties and values. One of the objects always has a subset of the properties set of the other object. For example, an object named Button_1 in the secondary object repository has the same description properties and values as an object named Button_1 in the primary object repository, but also has additional properties and values.

You can resolve this conflict type by:

- Keeping the object added from the primary object repository only.
- Keeping the object added from the secondary object repository only.
- Keeping the objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Edit_1.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

### Same Name Different Description Conflict

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

You can resolve this conflict type by:

- Keeping the object added from the primary object repository only.
- Keeping the object added from the secondary object repository only.
- Keeping the objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Edit_1.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

### Same Description Different Name Conflict

An object in the primary object repository and an object in the secondary object repository have different names, but the same description properties and values.

You can resolve this conflict type by:

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

## Resolving Object Conflicts

Conflicts between objects in the primary and secondary object repositories are resolved automatically by the Object Repository Merge Tool according to the default resolution settings that you can configure before performing the merge. For more information, see "Defining Default Settings" on page 261.

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

For example, an object in the primary object repository could have the same name as an object in the secondary object repository, but have a different description. You may have defined in the default settings that in this case, the object with the more generic object description, meaning the object with fewer properties, should be added to the target object repository. However, when you review the conflicts after the automatic merge, you could decide to handle the specific conflict differently, for example, by keeping both objects.

---

**Note:** Changes that you make to the default conflict resolution can themselves affect the target object repository by causing new conflicts. In the above example, keeping both objects would cause a name conflict. Therefore, the target object repository is updated after each conflict resolution change and redisplayed.

---

You can identify objects that caused conflicts, and the conflict type, by the icon displayed to the left of the object name in the target object repository pane of the Object Repository Merge Tool and the text color. When you select a conflicting object, a full description of the conflict, including how it was automatically resolved by the Object Repository Merge Tool, is displayed in the Resolutions Options pane.

The Resolutions Options pane offers alternative resolution options. You can choose to keep the default resolution if it suits your needs, or use the alternative options to resolve the conflict in a different way. In addition, for a local object repository merge, you can click the **Ignore Object** button to exclude a specific local object repository object from the target shared object repository.

---

**Tip:** You can also change the default resolution settings and merge the files again. For more information, see "Defining Default Settings" on page 261.

---

**To change the way in which object conflicts are resolved:**

**1** In the target object repository, select an object that had a conflict, as indicated by the icon to the left of the object name. The conflicting objects are highlighted in the source object repositories.

A description of the conflict and the resolution method used by the Object Repository Merge Tool is described in the Resolution Options pane. A radio button for each possible alternative resolution method is displayed. For information on each of the conflict types, see "Understanding Object Conflicts" on page 276.

2. In the Resolution Options pane, select a radio button to choose an alternative resolution method. The target object repository is updated according to your selection and redisplayed.
3. In the Resolution Options pane, click the **Previous Conflict** or **Next Conflict** buttons to jump directly to the next or previous conflict in the target object repository hierarchy.
4. Repeat steps 1 through 3 to modify additional conflict resolutions, as necessary.
5. Save the target object repository, as described in "Saving the Target Object Repository" on page 284.

## Filtering the Target Repository Pane

Merging two object repositories can result in a target object repository containing a large number of objects. To make navigation and the location of specific objects easier in the target object repository pane, the Object Repository Merge Tool enables you to filter the objects in the pane and show only the objects that had conflicts that were resolved during the merge.

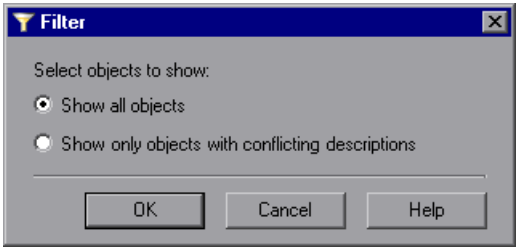
---

**Note:** The filter only affects which objects are displayed in the target object repository pane. It does not affect which objects are included in the target object repository.

---

**To filter the objects in the target object repository pane:**

**1** Choose **Tools** > **Filter** or click the **Filter** button. The Filter dialog box opens.




---


**Tip:** You can also click the **Filter** icon in the status bar to view the Filter dialog box. The Filter is shown as **ON** in the status bar when a filter is currently in use.

---

**2** Select a radio button according to the objects you want to view in the target object repository.

- **Show all objects.** Shows all objects in the target object repository
- **Show only objects with conflicting descriptions.** Shows only objects in the target object repository that have description conflicts

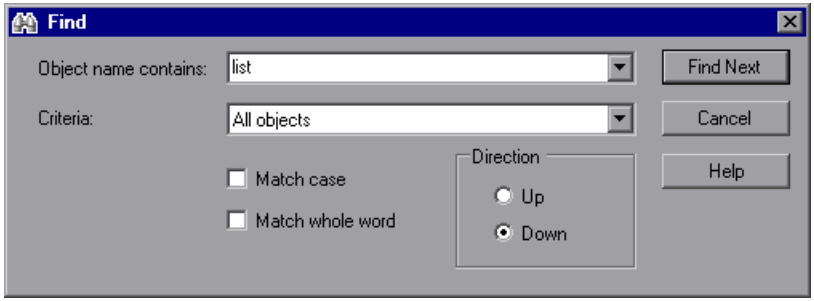
**3** Click **OK**. The objects in the pane are filtered and the target object repository displays only the requested object types. A progress bar is displayed in the status bar during the filter process.

# Finding Specific Objects

You can use the Find feature in the Object Repository Merge Tool to locate one or more objects in the target object repository whose name contains a specified string. The located object is also highlighted in the relevant primary and/or secondary object repositories.

**To find an object:**

**1** Choose **Navigate** > **Find** or click the **Find** button. The Find dialog box opens.




**2** In the **Object name contains** box, enter the full or partial name of the object you want to find.

**3** In the **Criteria** box, refine your search by selecting which objects to search. The following criteria are available:

- **All objects**
- **Objects from one source**
- **Objects with conflicts**
- **Objects with conflicts or from one source**

**4** Select one or both of the following options to help fine-tune your search:

- **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization exactly matches the text you entered in the **Object name contains** box.
- **Match whole word.** Searches for occurrences that are whole words only and not part of larger words.

5 Specify the direction from the current cursor location in which you want to search: **Up** or **Down**. The Find operation will continue to search the entire object repository after it reaches the beginning or end of the file.

6 Click **Find Next** to highlight the next object that matches the specified criteria in the target object repository.

You can also close the Find dialog box and use the following commands:

➤ Click the **Find Next** button or choose **Navigate** > **Find Next** to highlight the next object that matches the specified criteria.

➤ Click the **Find Previous** button or choose **Navigate** > **Find Previous** to highlight the previous object that matches the specified criteria.

## Saving the Target Object Repository

When you are sure that the object conflicts are resolved satisfactorily, you can save the target object repository to the file system or to a Quality Center project (if QuickTest is currently connected to the Quality Center project).

The file you can save depends on the types of object repositories that were merged. If you merged two shared object repositories, you can save the new target object repository file that was created. If you merged one or more local object repositories with a shared object repository, you can save the existing shared object repository file that now contains the objects and data from the local object repositories.

## Saving the Object Repository to the File System

You can save the new merged shared object repository to the file system at any time.

**To save an object repository to the file system:**

**1** Choose **File** > **Save** or click the **Save** button. If the file was saved previously, the current changes you made are saved. If the file has not yet been saved, the Save Shared Object Repository dialog box opens.

---

**Note:** If you are connected to Quality Center, the Save Shared Object Repository dialog box is different from the standard file selection dialog box. You can switch to save the file to the file system by clicking the **File System** button in that dialog box.

---

**2** Browse to and select the folder in which you want to save the object repository. Enter a name for the object repository in the **File name** box.

Use a descriptive name that will help you easily identify the file. You cannot use the following characters in an object repository name:
\ / : " ? < > | * '

**3** Click **Save**. QuickTest saves the object repository with a **.tsr** extension in the specified location and displays the file name and path above the target object repository in the Object Repository - Merge Tool window.

## Saving the Object Repository to a Quality Center Project

If you are connected to Quality Center, you can save your merged shared object repository as an attachment in the test plan tree of your project. Later, you can associate the object repository with the required application areas so that the objects in the object repository can be accessed by components. For more information, see *HP QuickTest Professional for Business Process Testing User's Guide*.

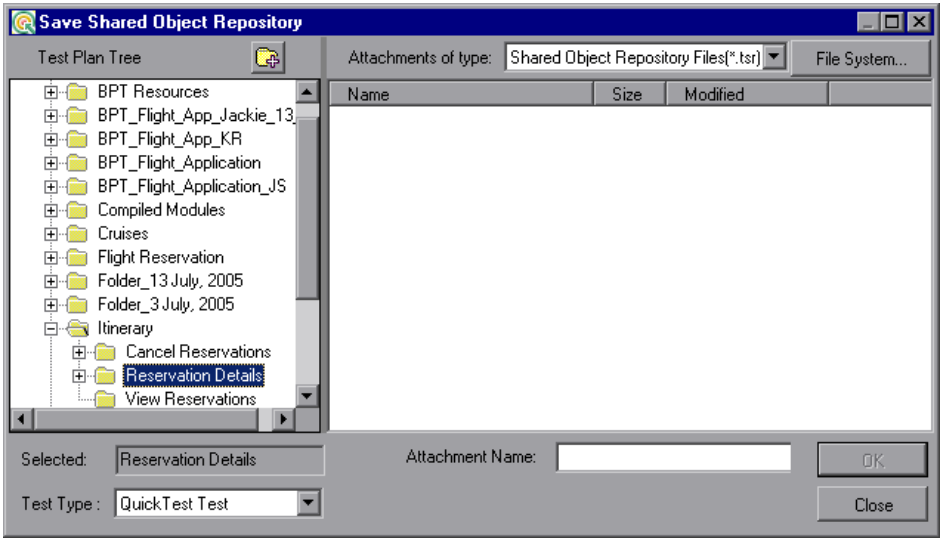
---

**Note:** You cannot overwrite an existing object repository in Quality Center.

---

**To save an object repository in a Quality Center project:**

**1** Choose **File** > **Save** or click the **Save** button. If the file was saved to Quality Center previously, the current changes you made are saved to the object repository. If the file has not yet been saved, the Save Shared Object Repository dialog box opens.



**2** In the test plan tree, select the folder in which you want to save the object repository.

You can also click the **New Folder** button to create a new test folder in the test plan tree in Quality Center.

---

**Note:** You can switch to save the file to the file system by clicking the **File System** button in the Save Shared Object Repository dialog box. You can switch back to the Save Shared Object Repository dialog box for Quality Center by clicking the **Quality Center** button.

---

**3** Enter a name for the object repository in the **Attachment Name** box.

Use a descriptive name that will help you easily identify the object repository. You cannot use the following characters in an object repository name:
\ / : " ? < > | * '

---

**Note:** You cannot overwrite an existing object repository.

---

**4** Click **OK**. QuickTest saves the object repository to Quality Center and displays the file name and path above the target object repository in the Object Repository - Merge Tool window. In Quality Center, the file is shown in the Attachments tab of the relevant folder.

# 8

# Comparing Shared Object Repositories

QuickTest Professional enables you to compare two shared object repositories using the Object Repository Comparison Tool, and view the differences in their objects, such as different object names, different object descriptions, and so on.

**This chapter includes:**

# About Comparing Shared Object Repositories

QuickTest Professional enables you to compare existing assets from two object repositories using the Object Repository Comparison Tool. The tool is accessible from the Object Repository Manager, and enables you to compare different object repository resources, or different versions of the same object repository resource, and identify similarities, variations, or changes.

Differences between objects in the two object repository files, named the **First** and **Second** files, are identified according to default rules. During the comparison process, the object repository files remain unchanged. For more information about the types of differences identified by the Object Repository Comparison Tool, see "Understanding Object Differences" on page 299.

After the compare process, the Comparison Tool provides a graphic presentation of the objects in the object repositories, which are shown as nodes in a hierarchy. Objects that have differences, as well as unique objects that are included in one object repository only, can be identified according to a color configuration that you can select. Objects that are included in one object repository only are identified in the other object repository by the text "Does not exist". You can also view the properties and values of each object that you select in either object repository.

You can use the information displayed by the Object Repository Comparison Tool when managing or merging object repositories. For more information, see Chapter 8, "Comparing Shared Object Repositories," or Chapter 7, "Merging Shared Object Repositories."

---

**Notes:**

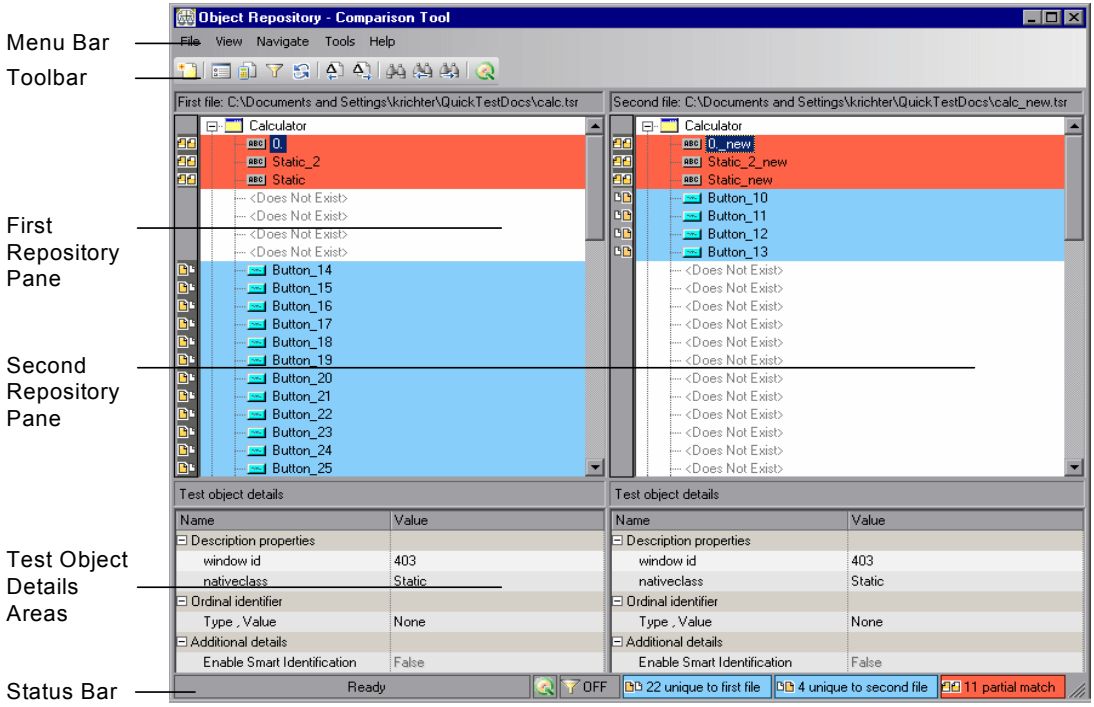
- The Object Repository Merge Tool does not merge checkpoint or output objects from a local object repository into the target shared object repository.
- You cannot work with the Object Repository Manager or the Object Repository Merge Tool when the Object Repository Comparison Tool is open.

---

# Understanding the Object Repository Comparison Tool

You open the Object Repository Comparison Tool by choosing **Tools** > **Object Repository Comparison Tool** in the Object Repository Manager.

An example window of the Object Repository - Comparison Tool is shown below:



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

- **Menu bar.** Displays menus of Object Repository Comparison Tool commands. These commands are described in various places throughout this chapter. Shortcut keys for menu commands are described in "Performing Object Repository Comparison Tool Commands Using Shortcut Keys" on page 296.
- **Toolbar.** Contains buttons of commonly used menu commands to assist you in comparing your object repositories and viewing the similarities and differences in their objects. Toolbar buttons are described in "Using Toolbar Commands" on page 295.
- **Repository Panes.** Display a hierarchical view of the objects in the object repositories being compared. In the column to the left of the object hierarchies, each pane displays icons representing the comparison of each object. For more information, see "Understanding the Repository Panes" on page 292.
- **Test Object Details areas.** Show the properties and values of the object selected in an object repository pane. For more information, see "Understanding the Repository Panes" on page 292.
- **Status Bar.** Shows the status of the comparison process and details of the differences found during the object repository comparison. For more information, see "Understanding the Status Bar" on page 294.

## Understanding the Repository Panes

The object repository panes display the hierarchies of the objects, and their properties and values, in the object repository files that you are comparing. The file path is shown above each object hierarchy.

To make it easier to see the status of an object at a glance, the text and background of object names in the object repositories are displayed using different colors, according to the type of difference found.

You can change the default colors used in the object repositories to indicate the difference type. For more information, see "Changing Color Settings" on page 300.

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

➤ Objects that are unique to the first file

➤ Objects that are unique to the second file

➤ Objects in both the first and second file that are not identical but partially match

For more information on all difference types, see "Understanding Object Differences" on page 299.

The object repository panes provide the following functionality:

➤ When you select an object in one object repository pane, the corresponding object in the other file hierarchy is located and highlighted. You can press the CTRL button when you select an object to highlight only the selected object without highlighting the corresponding object in the other file.

➤ When you select an object in an object repository pane, its properties and values are displayed in the respective **Test object details** area at the bottom of the pane.

➤ When you position your cursor over an icon to the left of an object in an object repository pane, the comparison details are displayed as a tooltip, for example, Partial match, or Unique to second file.

➤ You can expand or collapse the hierarchy of a parent node by double-clicking the node, or by clicking the expand (+) or collapse (-) symbol to the left of the node name. You can also expand or collapse the entire hierarchy in the object repository pane by choosing **Collapse All** or **Expand All** from the **View** menu.

➤ You can jump directly to the next or previous difference in the object repository hierarchy by choosing **Next Difference** or **Previous Difference** from the **Navigate** menu, by clicking the **Next Difference** or **Previous Difference** buttons in the toolbar, or by using keyboard shortcuts. For more information about shortcuts, see "Performing Object Repository Comparison Tool Commands Using Shortcut Keys" on page 296.

- You can locate one or more objects in the object repository panes by using the Find dialog box. For more information, see "Finding Specific Objects" on page 306.
- You can drag the edges of the panes to resize them in the Object Repository Comparison Tool window.

### Understanding the Status Bar

The status bar shows information about the comparison process and the results that are displayed:

- A progress bar is displayed on the left of the status bar during the comparison process. **Ready** is displayed when the process is complete.

- The Quality Center icon is displayed when QuickTest is connected to a Quality Center project.

- The filter status is shown next to the Filter icon: **OFF** indicates that the object repositories are not filtered and all objects are shown. **ON** indicates a filter is active and that some objects may have been filtered out of the display. You can click the **Filter** icon to view the Filter dialog box. For more information, see "Filtering the Repository Panes" on page 304.
- The number of differences found during the comparison are displayed, as follows:

  - The number of objects that are unique to the first file

  - The number of objects that are unique to the second file

  - The number of objects in the first and second file that are not identical but partially match

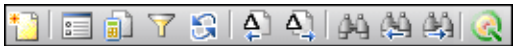
For more information on all difference types, see "Understanding Object Differences" on page 299.

# Using Object Repository Comparison Tool Commands

You can select Object Repository Comparison Tool commands from the menu bar or from the toolbar. You can also perform certain commands by pressing shortcut keys.

## Using Toolbar Commands

You can perform frequently used commands by clicking buttons in the toolbar.



| | Description |
|---|---|
|  | New Comparison (described in “File Menu Commands” on page 296) |
|  | Color Settings (described in “File Menu Commands” on page 296) |
|  | Statistics (described in “View Menu Commands” on page 297) |
|  | Filter (described in “Tools Menu Commands” on page 298) |
|  | Synchronized Nodes (described in “Navigate Menu Commands” on page 297) |
|  | Previous Difference (described in “Navigate Menu Commands” on page 297) |
|  | Next Difference (described in “Navigate Menu Commands” on page 297) |
|  | Find (described in “Navigate Menu Commands” on page 297) |
|  | Find Previous (described in “Navigate Menu Commands” on page 297) |
|  | Find Next (described in “Navigate Menu Commands” on page 297) |
|  | Quality Center Connection (described in “File Menu Commands” on page 296) |

## Performing Object Repository Comparison Tool Commands Using Shortcut Keys

You can perform frequently-used commands by clicking toolbar buttons or choosing the relevant menu option. You can also perform some commands by pressing the relevant shortcut keys.

### File Menu Commands

You can manage your object repository comparison using the following **File** menu commands:

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **New Comparison** | CTRL+N | Enables you to specify two object repositories on which to perform a new comparison operation. |
|  | **Quality Center Connection** | | Enables you to connect QuickTest to a Quality Center project. For more information, see "Connecting to Your Quality Center Project" on page 44. |
| | **Exit** | | Closes the Object Repository - Comparison Tool window. |

## View Menu Commands

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

| | Command | Function |
|---|---|---|
|  | **Statistics** | Opens the Statistics dialog box, which describes the comparison between the two repositories, including the number and type of any differences found. For more information, see "Viewing Comparison Statistics" on page 303. |
| | **Collapse All** | Collapses the entire hierarchy in both comparison panes.<br>**Tip:** Double-clicking an expanded node collapses it in both panes simultaneously. |
| | **Expand All** | Expands the entire hierarchy in both comparison panes.<br>**Tip:** Double-clicking a collapsed node expands it in both panes simultaneously. |

## Navigate Menu Commands

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

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Next Difference** | F4 | Finds the next difference between objects in the object repositories. |
|  | **Previous Difference** | SHIFT+F4 | Finds the previous difference between objects in the object repositories. |
|  | **Find** | CTRL+F | Opens the Find dialog box. |

| | Command | Shortcut Key | Function |
|---|---|---|---|
|  | **Find Next** | F3 | Finds the next object in the object repositories according to the search specifications in the Find dialog box. |
|  | **Find Previous** | SHIFT+F3 | Finds the previous object in the object repositories according to the search specifications in the Find dialog box. |

## Tools Menu Commands

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

| | Command | Function |
|---|---|---|
|  | **Synchronized Nodes** | Enables you to navigate the two object repository panes simultaneously or independently of one another. For more information, see "Synchronizing Object Repository Views" on page 305. |
|  | **Filter** | Opens the Filter dialog box, enabling you to specify the types of test object matches that you want to show. For more information, see "Filtering the Repository Panes" on page 304. |
|  | **Color Settings** | Opens the Settings dialog box, enabling you to specify the text color and background of the object names and empty nodes displayed in the comparison panes.<br>For more information, see "Changing Color Settings" on page 300. |

## Help Menu Command

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

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

# Understanding Object Differences

The Comparison Tool automatically identifies objects during the comparison process by classifying them into one of the following types:

- **Identical.** Objects that appear in both object repository files. There is no difference in their name or in their properties.
- **Matching description, different name.** Objects that appear in both object repository files that have different names, but the same description properties and values.
- **Similar description.** Objects that appear in both object repository files that have similar, but not identical, description properties and values. One of the objects always has a subset of the properties set of the other object. This implies that it is likely to be a less detailed description of the same object. For example, an object named Button_1 in the second object repository has the same description properties and values as an object named Button_1 in the first object repository, but also has additional properties and values.

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

---

**Note:** The Object Repository Comparison Tool gives precedence to matching object descriptions over the matching of object names. For this reason, certain object nodes may be linked during the comparison process and not others.

---

- **Unique to first file**, or **Unique to second file.** Objects that appear in only one of the object repository files.

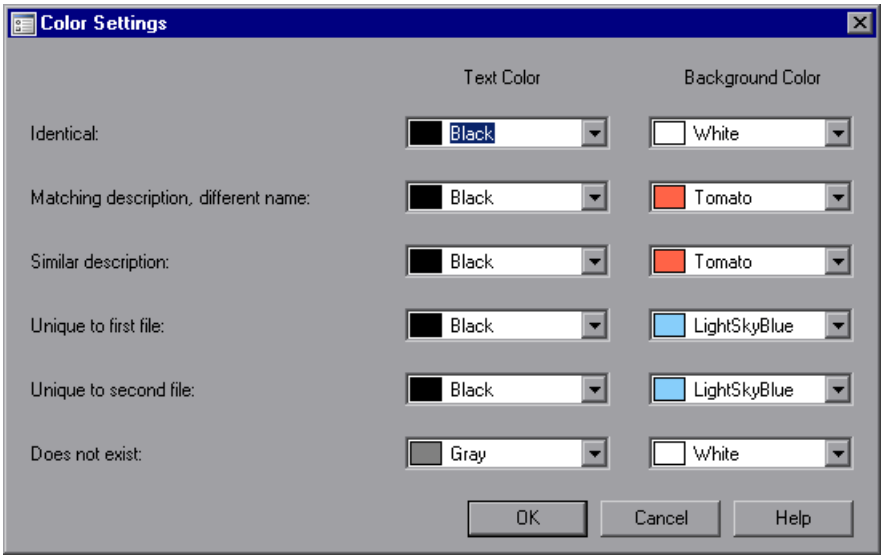
# Changing Color Settings

The text and background of object names, and empty nodes representing objects that exist in the other object repository only, are displayed in the Comparison Tool window in default colors, according to their difference types. This enables you to see the status of each object in the object repository panes. These text colors are also used in the Statistics dialog box.

You can change the default color settings if required.

**To change color settings:**

**1** Choose **Tools** > **Color Settings** or click the **Color Settings** button in the toolbar. The Color Settings dialog box opens.




**2** For each difference type, click the down arrow next to the text box and select an identifying text color and background color from the Custom, Web, or System tabs.

**3** Click **OK**. After performing a comparison of object repositories, object names and empty nodes in the respective object repository panes are displayed according to your selections.

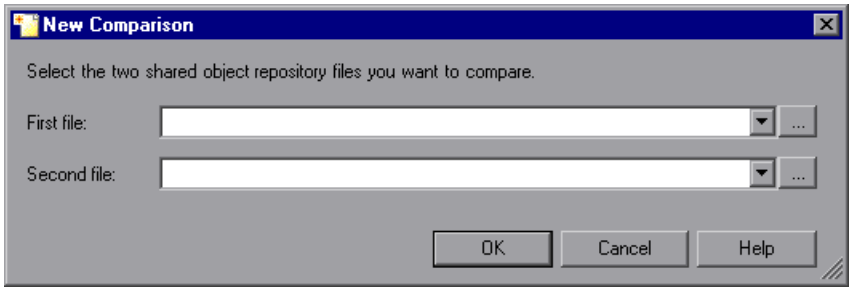
# Comparing Object Repositories

Using the Object Repository Comparison Tool, you can compare two object repositories according to predefined settings that define how differences between objects are identified.

**To compare two object repositories:**

1 In QuickTest Professional, choose **Resources** > **Object Repository Manager**.

2 In the Object Repository Manager, choose **Tools** > **Object Repository Comparison Tool**. The New Comparison dialog box opens on top of the Object Repository - Comparison Tool window.




---

**Tips:**

- If the Object Repository - Comparison Tool window is already open, you can choose **File** > **New Comparison** or click the **New Comparison** button in the toolbar to open the New Comparison dialog box.
- If you want to change the color settings before comparing the object repositories, click **Cancel** to close the New Comparison dialog box, change the settings as described in "Changing Color Settings" on page 300, and then perform the comparison.

---

**3** In the **First file** and **Second file** boxes, enter or browse to and select the **.tsr** object repository files that you want to compare. By default, the boxes display the last files selected for comparison using the Object Repository Comparison Tool. You can click the down arrow next to each box to view and select recently used files.

---

**Notes:**

- A warning icon is displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. You can position your pointer over the icon to see a tooltip explanation of the error. Enter or select an existing **.tsr** file with the correct path.
- If you want to compare an object repository that was created using a version of QuickTest earlier than version 9.0, you must first open and save it in the Object Repository Manager to update it to the new format.
- If you are connected to Quality Center, you can enter (or browse to) object repositories from Quality Center as well as from the file system.

---

**4** Click **OK**. The Object Repository Comparison Tool compares the objects in the selected object repositories and displays the results in the Statistics dialog box on top of the Object Repository - Comparison Tool window.

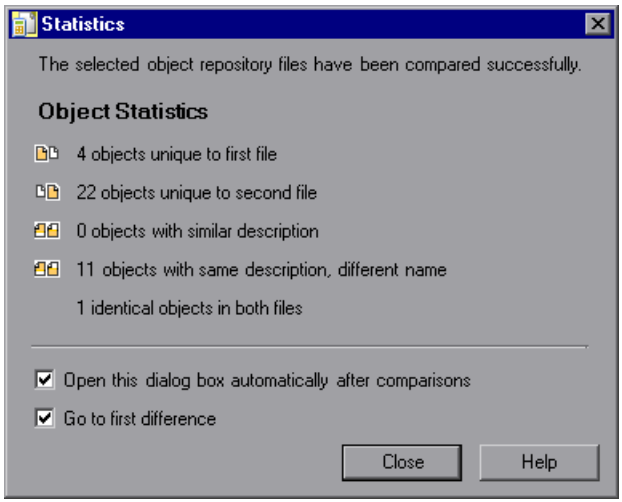
**5** Review the statistics, as described in "Viewing Comparison Statistics" on page 303, and click **Close**.

**6** In the Object Repository - Comparison Tool window, you can:

- Filter the objects in the object repositories, as described in "Filtering the Repository Panes" on page 304.
- Find specific objects in the object repositories, as described in "Finding Specific Objects" on page 306.

# Viewing Comparison Statistics

After you compare two object repositories, the Object Repository Comparison Tool displays the Statistics dialog box, which describes how the files were compared, and the number and type of any differences found.




---

**Tip:** You can choose not to view the Statistics dialog box every time you compare object repositories by clearing the **Open this dialog box automatically after comparisons** check box. You can view the comparison statistics in the Statistics dialog box at any time by choosing **View** > **Statistics** in the Comparison Tool window, or by clicking the **Statistics** button in the toolbar.

---

The Statistics dialog box displays the following information:

- The number and type of any differences between the objects in the object repositories. Difference types are described in "Understanding Object Differences" on page 299.
- The number of items that are unique to the first or the second file, or are identical in both files.

The icons displayed for each difference type in the object statistics are the same as those used in the object repository panes. For more information, see "Understanding the Repository Panes" on page 292.

---

**Tip:** Select the **Go to first difference** check box to jump to the first difference in the object repositories immediately after you close the Statistics dialog box.

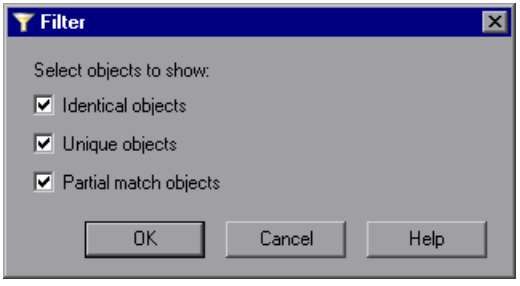
---

## Filtering the Repository Panes

Object repositories can contain a large number of objects. To make navigation and the location of specific objects easier in the object repository panes, the Object Repository Comparison Tool enables you to filter the objects and show only the objects that you want to view.

**To filter the objects in the object repository panes:**

**1** Choose **Tools** > **Filter** or click the **Filter** button in the toolbar. The Filter dialog box opens.




---


**Tip:** The **Filter** button in the toolbar is surrounded by a border when a filter is currently in use. In addition, the filter is shown as **ON** in the status bar. You can click the **Filter** icon in the status bar to open the Filter dialog box.

---

**2** Select one or more check boxes according to the objects you want to view in the object repositories.

- **Identical Objects.** Objects that appear in both object repository files and have no differences in their name or in their properties
- **Unique objects.** Objects that appear only in the first object repository file or only in the second object repository file
- **Partial match objects.** Objects in the object repository files that match but have name or description differences

---

**Tip:** Select all the check boxes to view all the objects in both object repositories.

---

**3** Click **OK**. The objects in the panes are filtered and the object repositories display only the requested object types.

## Synchronizing Object Repository Views

The Object Repository Comparison Tool enables you to navigate the two object repositories independently. You can also resize the various panes to display only some of the objects contained in the object repositories. When using large object repositories, this can result in the various panes displaying different areas of the object repository hierarchies, making it difficult to locate and track specific objects affected by the comparison process.

To synchronize the object repositories to display the same object in both views, select the object in the first or second object repository in which it is currently visible and click the **Synchronized Nodes** button in the toolbar. The matching node is highlighted in the other object repository and both object repositories scroll simultaneously.

**Tip:** The **Synchronized Nodes** button in the toolbar is surrounded by a border when the object repositories are currently synchronized. Click the **Synchronized Nodes** button again to navigate the two object repositories independently. When the object repositories are synchronized, you can also press the CTRL button while selecting an object to highlight the selected object only.

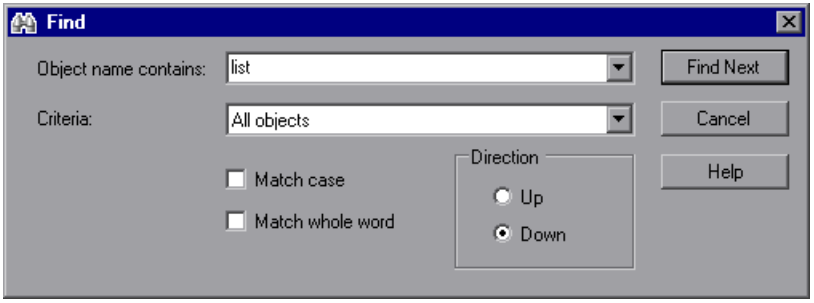
## Finding Specific Objects

You can use the Find feature in the Object Repository Comparison Tool to locate one or more objects in a selected object repository whose name contains a specified string. The located object is also highlighted in the other object repository if it exists there.

**To find an object:**

1 Click the object repository pane that contains the required object.

2 Choose **Navigate** > **Find** or click the **Find** button in the toolbar. The Find dialog box opens.




3 In the **Object name contains** box, enter the full or partial name of the object you want to find. You can click the down arrow  next to the box to view and select a recently used string.

**4** In the **Criteria** box, refine your search by selecting which objects to search. The following criteria are available:

- **All objects**
- **Unique objects**
- **Partial match objects**
- **Unique or partial match objects**

**5** Select one or both of the following options to help fine-tune your search:

- **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization exactly matches the text you entered in the **Object name contains** box.
- **Match whole word.** Searches for occurrences that are whole words only and not part of larger words.

**6** Specify the direction from the current cursor location in which you want to search: **Up** or **Down**. The Find operation will continue to search the entire file after it reaches the beginning or end of the object repository.

**7** Click the **Find Next** button to highlight the next object that matches the specified criteria in the object repository.

You can also close the Find dialog box and use the following commands:

- Click the **Find Next** button in the toolbar, choose **Navigate** > **Find Next**, or press F3, to highlight the next object that matches the specified criteria.

- Click the **Find Previous** button in the toolbar, choose **Navigate** > **Find Previous**, or press SHIFT+F3, to highlight the previous object that matches the specified criteria.

# Part III

## Defining Functions and Other Programming Tasks

# 9

# Working in Function Library Windows

You can use the QuickTest Function Library window to create function libraries using VBScript. This chapter provides a brief introduction to VBScript and shows you how to enhance your function libraries using a few simple programming techniques.

**This chapter includes:**

## About Working in the Function Library Window

You can create and work with function libraries using the Function Library window. To learn about working with VBScript, you can view the VBScript documentation directly from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

You can add statements that perform operations on objects or retrieve information from your application. For example, you can add a step that checks that an object exists, or you can retrieve the return value of a method.

You can add steps to your function library either manually or using the Step Generator. For more information on using the Step Generator, see the *HP QuickTest Professional User's Guide*.

You can print a function library at any time. You can also include additional information in the printout. For more information on printing a function library, see "Printing a Function Library" on page 384.

## Generating Statements in a Function Library

You can generate statements in the following ways:

- You can use the Step Generator to add steps that use methods and functions. For more information, see the *HP QuickTest Professional User's Guide*.
- You can manually insert VBScript statements that use methods to perform operations. QuickTest includes IntelliSense, a statement completion feature that helps you select the method or property for your statement and to view the relevant syntax as you type in a function library. For more information, see "Generating a Statement for an Object" on page 313.
- When you start to type a VBScript keyword in a function library, QuickTest automatically adds the relevant syntax or blocks to your script, if the **Auto-expand VBScript syntax** option is enabled. For more information, see "Automatically Completing VBScript Syntax" on page 315.

## Generating a Statement for an Object

When you type in a function library, IntelliSense (the statement completion feature included with QuickTest) enables you to select the method or property for your statement from a drop-down list and view the relevant syntax.

---

**Tip:** Although IntelliSense in function library documents is supported as described below 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.

---

The **Statement Completion** option is enabled by default. You can disable or enable this option in the Editor Options dialog box. For more information, see Chapter 10, "Customizing Function Library Windows."

When the **Statement Completion** option is enabled:

- If you type a period after a test object in a statement, QuickTest displays a list of the relevant methods, properties, and registered functions that you can add after the object you typed.
- If you type the name of a method or property, QuickTest displays a list of available methods and properties. Pressing CTRL+SPACE automatically completes the word if there is only one option, or highlights the first method or property (alphabetically) that matches the text you typed.
- If you type the name of a method or property, QuickTest displays the syntax for it, including its mandatory and optional arguments. When you add a step that uses a method or property, you must define a value for each mandatory argument associated with the method or property.

➤ If you press CTRL+SPACE, QuickTest displays a list of the relevant methods, properties, VBScript functions, user-defined functions, VBScript constants, and utility objects that you can add. If you use the Object property in your statement, if the object data is currently available or the open application, QuickTest displays native methods and properties of any run-time object in your application. For more information on the Object property, see "Accessing Run-Time Object Properties and Methods" on page 354.

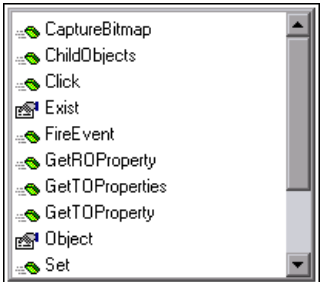
**To generate a statement using statement completion in a function library:**

**1** Confirm that the **Statement completion** option is selected (**Tools** > **View Options** > **General** tab).

**2** In the function library, type the full hierarchy of an object, for example:

```
Browser("Welcome: Mercury Tours").Page("Book a Flight:
    Mercury).WebEdit("username").
```

**3** Type a period (.) after the object description, for example ("username"). QuickTest displays a list of the available methods and properties for the object.

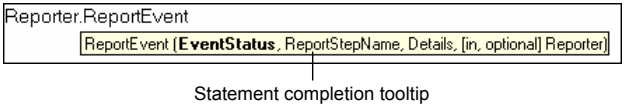



---

**Tip:** You can press CTRL+SPACE or choose **Edit** > **Advanced** > **Complete Word** after a period, or after you have begun to type a method or property name. QuickTest automatically completes the method or property name if only one method or property matches the text you typed. If more than one method or property matches the text, the first method or property (alphabetically) that matches the text you typed is highlighted.

---

**4** Double-click a method or property in the list or use the arrow keys to choose a method or property and press ENTER. QuickTest inserts the method or property into the statement. If the method or property contains arguments, QuickTest displays the syntax of the method or property in a tooltip.




In the above example, the ReportEvent method has four arguments.

---

**Tip:** You can also place the cursor on any method or function that contains arguments and press CTRL+SHIFT+SPACE or choose **Edit** > **Advanced** > **Argument Info** to display the statement completion (argument syntax) tooltip for that item.

---

**5** Enter the method arguments after the method.

For more details and examples of any QuickTest method, see the *HP QuickTest Professional Object Model Reference*.

For more information on VBScript syntax, see "Understanding Basic VBScript Syntax" on page 325.

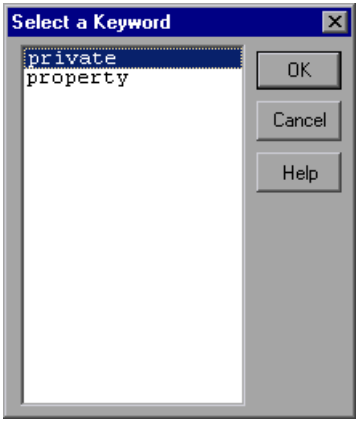
### Automatically Completing VBScript Syntax

When the **Auto-expand VBScript syntax** option is enabled and you start to type a VBScript keyword in a function library, QuickTest automatically recognizes the first two characters of the keyword and adds the relevant VBScript syntax or blocks to the script. For example, if you enter the letters if and then enter a space at the beginning of a line, QuickTest automatically enters:

```
If Then
End If
```

The **Auto-expand VBScript syntax** option is enabled by default. You can disable or enable this option in the Editor Options dialog box. For more information, see "Customizing Editor Behavior" on page 364.

If you enter two characters that are the initial characters of multiple keywords, the Select a Keyword dialog box is displayed and you can select the keyword you want. For example, if you enter the letters pr and then enter a space, the Select a Keyword dialog box opens containing the keywords private and property.




You can then select a keyword from the list and click **OK**. QuickTest automatically enters the relevant VBScript syntax or block in the script.

For more information on VBScript syntax, see "Understanding Basic VBScript Syntax" on page 325.

# Navigating in Function Libraries

You can use the Go To dialog box or bookmarks to jump to a specific line in a function library. You can also find specific text strings in a function library, and, if desired, replace them with different strings. These options make it easier to navigate through sections of a long function.

## Using the Go To Dialog Box

You can use the Go To dialog box to navigate to a specific line in a function library.

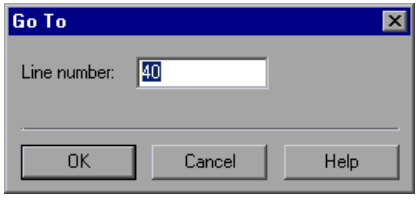
---

**Tip:** By default, line numbers are displayed in function libraries. If they are not displayed, you can select the **Show line numbers** option in the **Tools** > **View Options** > **General** tab. For more information on the Editor options, see Chapter 10, "Customizing Function Library Windows."

---

**To navigate to a line in a function library using the Go To dialog box:**

1. Activate the function library, if needed.
2. Select **Edit** > **Go To**. The Go To dialog box opens.



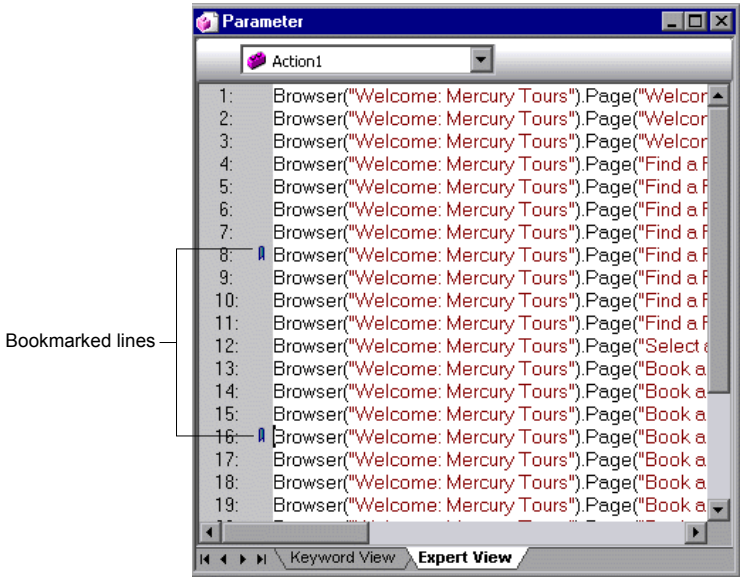

3. Enter the line to which you want to navigate in the **Line number** box and click **OK**. The cursor moves to the beginning of the line you specify.

## Working with Bookmarks

You can use bookmarks to mark important sections in your function library so that you can navigate between the various parts more easily. Bookmarks are not preserved when you navigate between documents, and they are not saved with the function library.

When you assign a bookmark, an icon is added to the left of the selected line in the function library. You can then use the **Go To** button in the Bookmarks dialog box to jump to the bookmarked rows.

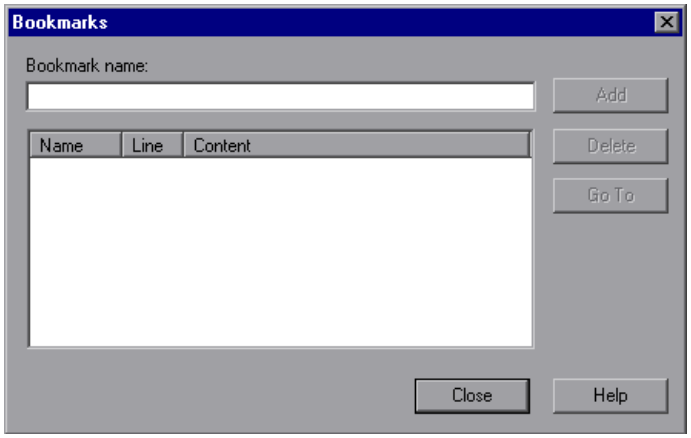
Bookmarks look the same in tests and in function libraries. In the following example, two bookmarks have been added to an action in a test.




**To set bookmarks:**

1. Activate the function library, if needed.
2. Click in the line to which you want to assign a bookmark.

**3** Choose **Edit** > **Bookmarks**. The Bookmarks dialog box opens.




**4** In the **Bookmark name** field, enter a unique name for the bookmark and click **Add**. The bookmark is added to the Bookmarks dialog box, together with the line number at which it is located and the textual content of the line. In addition, a bookmark icon is added to the left of the selected line in the function library.

**5** To delete a bookmark, select it in the list and click **Delete**.

**To navigate to a specific bookmark:**

**1** Activate the function library, if needed.

**2** Choose **Edit** > **Bookmarks**. The Bookmarks dialog box opens.

**3** Select a bookmark from the list and click the **Go To** button. QuickTest jumps to the appropriate line in the function library.

## Finding Text Strings

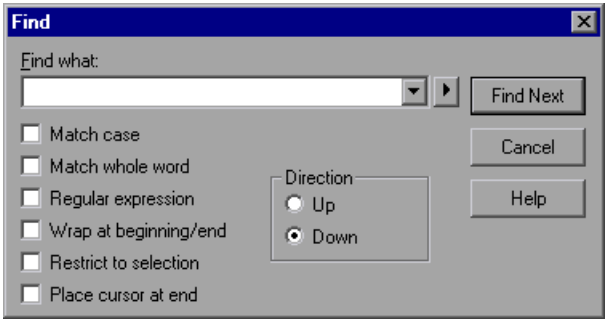
You can specify text strings to locate in a function library. You can either search for literal text or use regular expressions for a more advanced search. You can also use other options to further fine-tune your search results.

**To find a text string:**

**1** In the function library, perform one of the following:

- Click the **Find** button.
- Choose **Edit** > **Find**.

The Find dialog box opens.




**2** In the **Find what** box, enter the text string you want to locate.

**3** If you want to use regular expressions in the string you specify, click the arrow button ( ) and select a regular expression. When you select a regular expression from the list, it is automatically inserted in the **Find what** box at the cursor location. For more information, see "Using Regular Expressions in the Find and Replace Dialog Boxes" on page 323.

**4** Select any of the following options to help fine-tune your search:

- **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization matches the text you entered in the **Find what** box exactly.
- **Match whole word.** Searches for occurrences that are only whole words and not part of longer words.
- **Regular expression.** Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.

- **Wrap at beginning/end.** Continues the search from the beginning or end of the function library text when either the beginning or end is reached, depending on the selected search direction.
- **Restrict to selection.** Searches only within the selected part of the function library text.
- **Place cursor at end.** Places the cursor at the end of the highlighted occurrence when the search string is located.

**5** Specify the direction in which you want to search, from the current cursor location in the function library: **Up** or **Down**

**6** Click **Find Next** to highlight the next occurrence of the specified string in the active function library.

## Replacing Text Strings

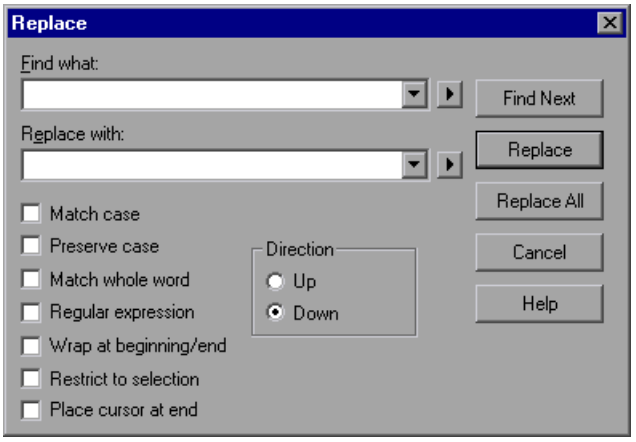
You can specify text strings to locate in the current function library, and specify the text strings you want to use to replace them. You can either find and replace literal text or use regular expressions for a more advanced process. You can also use other options to further fine-tune your find and replace process.

**To replace a text string:**

**1** In the function library, perform one of the following:

- Click the **Replace** button.
- Choose **Edit** > **Replace**.

The Replace dialog box opens.




**2** In the **Find what** box, enter the text string you want to locate.

**3** In the **Replace with** box, enter the text string you want to use to replace the found text.

**4** If you want to use regular expressions in the **Find what** or **Replace with** string, click the arrow button ( ▸ ) and select a regular expression. When you select a regular expression from the list, it is automatically inserted in the **Find what** or **Replace with** box at the cursor location. For more information, see "Using Regular Expressions in the Find and Replace Dialog Boxes" on page 323.

**5** Select any of the following options to help fine-tune your search:

- **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization exactly matches the text you entered in the **Find what** box.
- **Preserve case.** Checks each occurrence of the **Find what** string for all lowercase, all uppercase, sentence caps or mixed case. The **Replace with** string is converted to the same case as the occurrence found, except when the occurrence found is mixed case. In this case, the **Replace with** string is used without modification.
- **Match whole word.** Searches for occurrences that are whole words only and not part of longer words.

- **Regular expression.** Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.
- **Wrap at beginning/end.** Continues the search from the beginning or end of the function library text when either the beginning or end is reached, depending on the selected search direction.
- **Restrict to selection.** Searches only within the selected part of the function library text.
- **Place cursor at end.** Places the cursor at the end of the highlighted occurrence when the search string is located.
- **Direction.** Specifies the search direction.
  - **Up.** Searches only from the current text up to the beginning of the function library text.
  - **Down.** Searches only from the current text down to the end of the function library text.

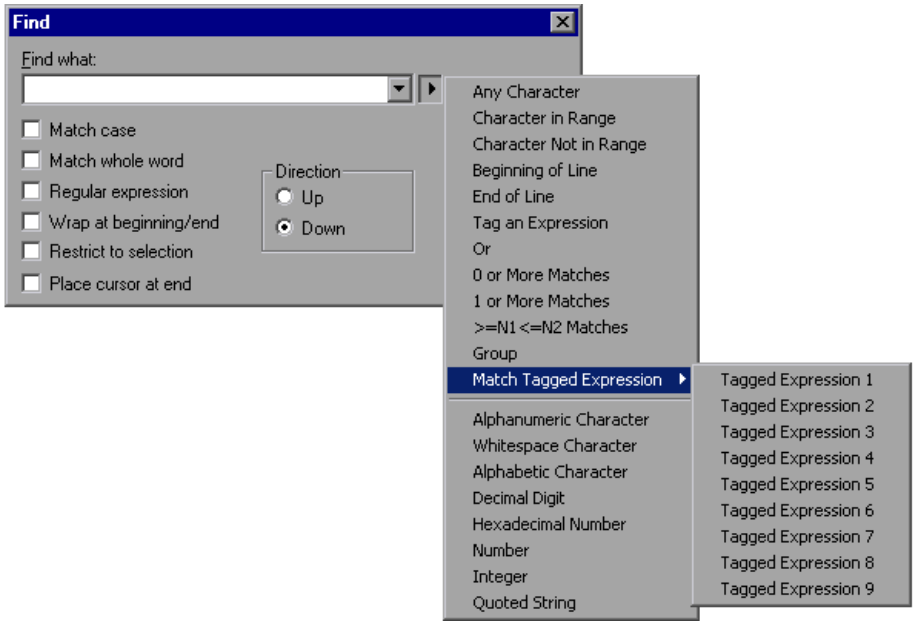
**6** Click **Find Next** to highlight the next occurrence of the specified text string in the active function library.

**7** Click **Replace** to replace the highlighted text with the text in the **Replace with** box, or click **Replace All** to replace all occurrences specified in the **Find what** box with the text in the **Replace with** box in the active function library.

### Using Regular Expressions in the Find and Replace Dialog Boxes

You can use regular expressions in the **Find what** and **Replace with** strings to enhance your search. For a general understanding of regular expressions, see "Understanding and Using Regular Expressions" on page 734. Note that there are differences in the expressions supported by the Find and Replace dialog boxes and the expressions supported in other parts of QuickTest.

You display the regular expressions available for selection by clicking the arrow button in the Find or Replace dialog boxes.




You can select from a predefined list of regular expressions. You can also use tagged expressions. When you use regular expressions to search for a string, you may want the string to change depending on what was already found.

For example, you can search for **(save\:n)\1**, which will find any occurrence of **save** followed by any number, immediately followed by **save**, as well as the same number that was already found (meaning that it will find **save6save6** but not **save6save7**).

You can also use tagged expressions to insert parts of what is found into the replace string. For example, you can search for **save(\:n)** and replace it with **open\1**. This will find **save** followed by any number, and replace it with **open** and the number that was found.

Select **Tag an Expression** from the regular expressions list to insert parentheses "**()**" to indicate a tagged expression in the search string.

Select **Match Tagged Expression** and then select the specific tag group number to specify the tagged expression you want to use, in the format '\' followed by a tag group number 1-9. (Count the left parentheses '**(**' in the search string to determine a tagged expression number. The first (left-most) tagged expression is "**1**" and the last is "**9**".)

## Understanding Basic VBScript Syntax

You write function libraries using VBScript, a powerful scripting language.

This section provides some basic guidelines to help you use VBScript statements to enhance your QuickTest function library. For more detailed information on using VBScript, you can view the VBScript documentation from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

Each VBScript statement has its own specific syntax rules. If you do not follow these rules, errors will be generated when you run the problematic step. You can check the syntax of the function library script at any time by clicking the **Check Syntax** button, or by choosing **Tools** > **Check Syntax**.

When working in a function library, you should consider the following general VBScript syntax rules and guidelines:

- **Case-sensitivity.** By default, VBScript is not case sensitive and does not differentiate between upper-case and lower-case spelling of words, for example, in variables, object and method names, or constants.

  For example, the two statements below are identical in VBScript:

  ```
  Browser("Mercury").Page("Find a Flight:").WebList("toDay").Select "31"
  browser("mercury").page("find a flight:").weblist("today").select "31"
  ```

- **Text strings.** When you enter a value as a text string, you must add quotation marks before and after the string. For example, in the above segment of script, the names of the Web site, Web page, and edit box are all text strings surrounded by quotation marks.

  Note that the value 31 is also surrounded by quotation marks, because it is a text string that represents a number and not a numeric value.

In the following example, only the property name (first argument) is a text string and is in quotation marks. The second argument (the value of the property) is a variable and therefore does not have quotation marks. The third argument (specifying the timeout) is a numeric value, which also does not need quotation marks.

```
Browser("Mercury").Page("Find a Flight:").WaitProperty("items count",
    Total_Items, 2000)
```

- **Variables.** You can specify variables to store strings, integers, arrays and objects. Using variables helps to make your script more readable and flexible. For more information, see "Using Variables" on page 327.
- **Parentheses.** To achieve the desired result and to avoid errors, it is important that you use parentheses () correctly in your statements. For more information, see "Using Parentheses" on page 328.
- **Indentation.** You can indent or outdent your script to reflect the logical structure and nesting of the statements. For more information, see "Formatting VB Script Text" on page 329.
- **Comments.** You can add comments to your statements using an apostrophe ('), either at the beginning of a separate line, or at the end of a statement. It is recommended that you add comments wherever possible, to make your scripts easier to understand and maintain. For more information, see "Formatting VB Script Text" on page 329, and "Inserting Comments" on page 345.
- **Spaces.** You can add extra blank spaces to your script to improve clarity. These spaces are ignored by VBScript.

For more information on using specific VBScript statements to enhance your function libraries, see "Using Comments, Control-Flow, and Other VBScript Statements" on page 345.

## Using Variables

You can specify variables to store test objects or simple values in your 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).

---

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 function library. In the following example, the Dim statement is used to declare the actual_value variable, which can then be used in different statements within the current function library:

```
Dim actual_value
    ' Get the actual property value
    actual_value = obj.GetROProperty(PropertyName)
```

## Using Parentheses

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 a function.

---

**Tip:** If you receive an **Expected end of statement** error message when running a step in your function library, it may indicate that you need to add parentheses around the arguments of the step's method.

---

Following are several examples showing when to use or not use parentheses.

The following example requires parentheses around the method arguments for the ChildItem method because it returns a value to a variable.

```
Set WebEditObj = Browser("Mercury Tours").Page("Method of Payment").
    WebTable("FirstName").ChildItem (8, 2, "WebEdit", 0)
WebEditObj.Set "Example"
```

The following example requires parentheses around the method arguments because Call is being used.

```
Call 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 kind of").
    WaitProperty("attribute/readyState", "complete", 4) Then
    Browser("index").Page("index").Link("All kind of").Click
End If
```

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
```

## Formatting VB Script Text

When working in a function library, it is important to follow accepted VBScript practices for comments and indentation.

Use comments to explain sections of a script. This improves readability and make function libraries easier to maintain and update. For more information, see "Inserting Comments" on page 345.

Use indentation to reflect the logical structure and nesting of your statements.

➤ **Adding Comments.** You can add comments to your statements by adding an apostrophe ('), either at the beginning of a separate line, or at the end of a statement.

---

**Tips:**

➤ You can comment a statement by clicking anywhere in the statement and clicking the **Comment Block** button.

➤ You can comment a selected block of text by clicking the **Comment Block** button, or by choosing **Edit** > **Advanced** > **Comment Block**. Each line in the block will be preceded by an apostrophe.

---

➤ **Removing Comments.** You can remove comments from your statements by deleting the apostrophe ('), either at the beginning of a separate line, or at the end of a statement.

**Tip:** You can remove the comments from a selected block or line of text by clicking the **Uncomment Block** button, or by choosing **Edit** > **Advanced** > **Uncomment Block**.

- **Indenting Statements.** You can indent your statements by selecting the text and choosing **Edit** > **Advanced** > **Indent** or by press the TAB key. The text is indented according to the tab spacing selected in the Editor Options dialog box, as described in "Customizing Editor Behavior" on page 364.

**Note:** The **Indent selected text when using the Tab key** check box must be selected in the Editor Options dialog box, otherwise pressing the TAB key will delete the selected text.

- **Outdenting Statements.** You can outdent your statements by selecting **Edit** > **Advanced** > **Outdent** or by deleting the space at the beginning of the statements.

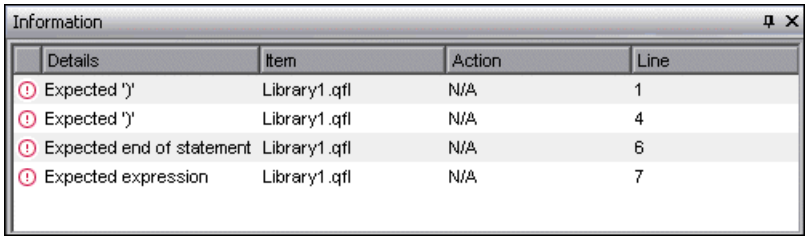
For more detailed information on formatting in VBScript, you can view the VBScript documentation from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

## Handling VBScript Syntax Errors

You can check the syntax of the current function library at any time by clicking the **Check Syntax** button, or by choosing **Tools** > **Check Syntax**. If QuickTest finds any errors, it displays them in the Information pane.

You can view a description of each of the VBScript errors in the VBScript Reference. For more information, choose **Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript** > **Reference** > **Errors** > **VBScript Syntax Errors**.

The Information pane lists the syntax errors found in your document, and enables you to locate each syntax error so that you can correct it.




The Information pane shows the following information for each syntax error:

| Pane Element | Description |
|---|---|
| **Details** | The description of the syntax error. For example, if you opened a conditional block with an If statement but did not close it with an End If statement, the description is Expected 'End If'.<br>**Note:** In certain cases, QuickTest is unable to identify the exact error and displays a number of possible error conditions, for example: Expected 'End Sub', or 'End Function', or 'End Property'. Check the statement at the specified line to clarify which error is relevant in your case. |
| **Item** | The name of the function library containing the problematic statement. |
| **Action** | This column is not relevant for function libraries that are associated with business components (via application areas). |
| **Line** | The line containing the syntax error. Lines are numbered from the beginning of each function library. |

### Using the Information Pane

➤ Move the pointer over the description of a syntax error to display the currently incorrect syntax.

➤ To navigate to the line containing a specific syntax error, double-click the syntax error in the Information pane.

- You can resize the columns in the Information pane to make the information more readable by dragging the column headers.
- You can sort the details in the Information pane in ascending or descending order by clicking the column header.
- You can press F1 on an error in the Information pane to display information about VBScript syntax errors.

## Using Programmatic Descriptions

When QuickTest learns an object in your application, it adds the appropriate test object to the object repository. After the object exists in the object repository, you can add statements in the Expert View to perform additional methods 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 method.

For example, in the statement below, username is the name of an edit box. The edit box is located on a page with the name Mercury Tours, and the page exists in a browser with the name Mercury Tours.

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username")
```

Because each object in the object repository has a unique name, the object name is all you need to specify. During the run session, QuickTest finds the object in the object repository based on its name and parent objects, and uses the stored test object description for that test object to identify the object in your application.

You can also instruct QuickTest to perform methods on objects without referring to the object repository or to the object's name. To do this, you provide QuickTest with a list of properties and values that QuickTest can use to identify the object or objects on which you want to perform a method.

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 Test 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.


For example, suppose you are testing a Web site that generates a list of potential employers based on biographical information you provide, and offers to send your resume to the employer names you select from the list. You want your test to select all the employers displayed in the list, but when you design your test, you do not know how many check boxes will be displayed on the page, and you cannot, of course, know the exact object description of each check box. In this situation, you can use a programmatic description to instruct QuickTest to perform a Set "ON" method for all objects that fit the description: HTML TAG = input, TYPE = check box.

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.
- **Dynamic.** You add a collection of properties and values to a Description object, and then enter the Description object name in the statement.

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.

## Entering Programmatic Descriptions Directly into Statements

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 test object property and its value. Each property:=value pair should be separated by commas and quotation marks.

Note that 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)
```

---

**Note:** QuickTest evaluates all property values in programmatic descriptions as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *, ?, or +), use the \ (backslash) character to instruct QuickTest to treat the special characters as literal characters.

---

The statement below specifies a WebEdit test object in the Mercury Tours page with the Name author and an index of 3. During the run session, QuickTest finds the WebEdit object with matching property values and enters the text Mark Twain.

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("Name:=Author",
    "Index:=3").Set "Mark Twain"
```

---

**Note:** 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 specified using programmatic descriptions, QuickTest cannot identify the object.

For example, 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"
```

QuickTest tries to locate the WebEdit object based on its name, but cannot locate it in the repository because the parent objects were specified using programmatic descriptions.

---

For more information on working with test objects, see Chapter 4, "Working with Objects."

If you want to use the same programmatic description several times in a function library, you may want to assign the object you create to a variable.

For example, instead of entering:

```
Window("Text:=Myfile.txt - Notepad").Move 50, 50
Window("Text:=Myfile.txt - Notepad").WinEdit("AttachedText:=Find what:").
    Set "hello"
Window("Text:=Myfile.txt - Notepad").WinButton("Caption:=Find next").Click
```

You can enter:

```
Set MyWin = Window("Text:=Myfile.txt - Notepad")
MyWin.Move 50, 50
MyWin.WinEdit("AttachedText:=Find what:").Set "hello"
MyWin.WinButton("Caption:=Find next").Click
```

## Using Description Objects for Programmatic Descriptions

You can use the Description object to return a Properties collection object containing a set of Property objects. A Property object consists of a property name and value. You can then specify the returned Properties collection in place of an object name in a statement. (Each property object contains a property name and value pair.)

---

**Note:** By default, the value of all Property objects added to a Properties collection are treated as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *****, ?, +), use the \ (backslash) character to instruct QuickTest to treat the special characters as literal characters.

You can set the RegularExpression property to False to specify a value as a literal value for a specific Property object in the collection. For more information, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

---

To create the Properties collection, you enter a **Description.Create** statement using the following syntax:

**Set** *MyDescription* = **Description.Create()**

Once you have created a Properties object (such as MyDescription in the example above), you can enter statements to add, edit, remove, and retrieve properties and values to or from the Properties object during the run session. This enables you to determine which, and how many properties to include in the object description in a dynamic way during the run session.

After you fill the Properties collection with a set of Property objects (properties and values), you can specify the Properties object in place of an object name in a test statement.

For example, instead of entering:

```
Window("Error").WinButton("text:=OK", "width:=50").Click
```

you can enter:

```
Set MyDescription = Description.Create()
MyDescription("text").Value = "OK"
MyDescription("width").Value = 50
Window("Error").WinButton(MyDescription).Click
```

---

**Note:** When using programmatic descriptions from a specific point within a test object hierarchy, you must continue to use programmatic descriptions from that point onward within the same statement. If you specify a test object by its object repository name after other objects in the hierarchy have been described using programmatic descriptions, QuickTest cannot identify the object.

For example, you can use Browser(Desc1).Page(Desc1).Link(desc3), since it uses programmatic descriptions throughout the entire test object hierarchy.

You can also use Browser("Index").Page(Desc1).Link(desc3), since it uses programmatic descriptions from a certain point in the description (starting from the Page object description).

However, you cannot use Browser(Desc1).Page(Desc1).Link("Example1"), since it uses programmatic descriptions for the Browser and Page objects but then attempts to use an object repository name for the Link test object (QuickTest tries to locate the Link object based on its name, but cannot locate it in the repository because the parent objects were specified using programmatic descriptions).

---

When working with Properties objects, you can use variable names for the properties or values to generate the object description based on properties and values you retrieve during a run session.

You can create several Properties objects in your test if you want to use programmatic descriptions for several objects.

For more information on the Description and Properties objects and their associated methods, see the *HP QuickTest Professional Object Model Reference*.

## Retrieving Child Objects

You can use the ChildObjects method to retrieve all objects located inside a specified parent object, or only those child objects that fit a certain programmatic description. To retrieve this subset of child objects, you first create a description object and 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.

---

Once you have "built" a description in your description object, use the following syntax to retrieve child objects that match the description:

**Set** *MySubSet*=*TestObject*.**ChildObjects(***MyDescription***)**

For example, the statements below instruct QuickTest to select all of the check boxes on the Itinerary Web page:

```
Set MyDescription = Description.Create()
MyDescription("html tag").Value = "INPUT"
MyDescription("type").Value = "checkbox"

Set Checkboxes =
Browser("Itinerary").Page("Itinerary").ChildObjects(MyDescription)
NoOfChildObjs = Checkboxes.Count
For Counter=0 to NoOfChildObjs-1
    Checkboxes(Counter).Set "ON"
Next
```

In the Test Results, square brackets around a test object name indicate that the test object was created dynamically during the run session using the ChildObjects method or a programmatic description.


For more information on the ChildObjects method, see the *HP QuickTest Professional Object Model Reference*.

### Using the Index Property in Programmatic Descriptions

The index property can sometimes be a useful test object property for uniquely identifying an object. The **index** test object property identifies an object based on the order in which it appears within the source code, where the first occurrence is 0.

Index property values are object-specific. Thus, if you use an index value of 3 to describe a WebEdit test object, QuickTest searches for the fourth WebEdit object in the page.

If you use an index value of 3 to describe a WebElement object, however, QuickTest searches for the fourth Web object on the page regardless of the type, because the WebElement object applies to all Web objects.

For example, suppose you have a page with the following objects:

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

The description below refers to the third item in the list above, as it is the first WebEdit object on the page with the name UserName:

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

The following description, however, refers to the second item in the list above, as that is the first object of any type (WebElement) with the name UserName.

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

---

**Note:** If there is only one object, using index=0 will not retrieve it. You should not include the **index** property in the object description.

---

## Performing Programmatic Description Checks

You can compare the run-time value of a specified object property with the expected value of that property using either programmatic descriptions or user-defined functions.

Programmatic description checks are useful in cases in which you cannot apply a regular checkpoint, for example, if the object whose properties you want to check is not stored in an object repository. You can then write the results of the check to the Test Results report.

For example, suppose you want to check the run-time value of a Web button. You can use the GetROProperty or Exist methods to retrieve the run-time value of an object or to verify whether the object exists at that point in the run session.

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 method:

```
ActualDisabledVal =
Browser(micClass:="Browser").Page(micClass:="Page").WebButton
    (alt:=Continue").GetROProperty("disabled")
```

Using the Exist method:

```
While Not Browser(micClass:="Browser").Page(micClass:="Page").WebButton
    (alt:=Continue").Exist(30)
Wend
```

By adding Report.ReportEvent statements, you can instruct QuickTest to send the results of a check to the Test Results.

```
If ActualDisabledVal = True Then
Reporter.ReportEvent micPass, "CheckContinueButton = PASS", "The
Continue
    button is disabled, as expected."
Else
Reporter.ReportEvent micFail, "CheckContinueButton = FAIL", "The Continue
    button is enabled, even though it should be disabled."
```

You can also create and use user-defined functions to check whether your application is functioning as expected. The following example illustrates a function that checks whether an object is disabled and returns **True** if the object is disabled:

```
'@Description Checks whether the specified test object is disabled
'@Documentation Check whether the <Test object name> <test object type> is
enabled.
Public Function VerifyDisabled (obj)
    Dim enable_property
    ' Get the disabled property from the test object
    enable_property = obj.GetROProperty("disabled")
    If enable_property = 1 Then ' The value is True (1)—the object is disabled
        Reporter.ReportEvent micPass, "VerifyDisabled Succeeded", "The test
object is disabled, as expected."
        VerifyDisabled = True
    Else
        Reporter.ReportEvent micFail, "VerifyDisabled Failed", "The test object is
enabled, although it should be disabled."
        VerifyDisabled = False
    End If
End Function
```

---

**Note:** For information on using the GetROProperty method, see "Retrieving Run-Time Object Properties" on page 354. For information on using While...Wend statements, see "While...Wend Statement" on page 350. For information on specific test objects, methods, and properties, see the *HP QuickTest Professional Object Model Reference*.

---

# Running and Closing Applications Programmatically

You can run any application from a specified location using a SystemUtil.Run statement in a function library. This is especially useful if you want to provide an operation (function) that opens an application from within a component. You can specify an application and pass any supported parameters, or you can specify a file name and the associated application starts with the specified file open.

You can close most applications using the Close method. You can also use SystemUtil statements to close applications. For more information, see the *HP QuickTest Professional Object Model Reference*.

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
```

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

# Using Comments, Control-Flow, and Other VBScript Statements

QuickTest enables you to incorporate decision-making into your function library by adding conditional statements that control the logical flow of your function library. In addition, you can define messages in your test that QuickTest sends to your test results. To improve the readability of your function libraries, you can also add comments to them.

---

**Note:** The **VBScript Reference** (available from **Help** > **QuickTest Professional Help**) contains Microsoft VBScript documentation, including VBScript, Script Runtime, and Windows Script Host.

---

## Inserting Comments

A comment is a line or part of a line in a script that is preceded by an apostrophe ('). When you run a test, QuickTest does not process comments. Use comments to explain sections of a script to improve readability and to make function libraries easier to update.

The following example shows how a comment describes the purpose of the statement below it:

```
'Sets the word "mercury" into the "username" edit box.
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").
    Set "mercury"
```

By default, comments are displayed in green in function libraries. You can customize the appearance of comments in the Editor Options dialog box. For more information, see "Customizing Element Appearance" on page 367.

---

**Tips:**

➤ You can comment a block of text by choosing **Edit** > **Advanced** > **Comment Block** or by clicking the **Comment Block** button.

➤ To remove the comment, choose **Edit** > **Advanced** > **Uncomment Block** or click the **Uncomment Block** button.

---

---

**Note:** You can also add a comment line using the VBScript Rem statement. For more information, see the Microsoft VBScript Language Reference (choose **Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript**).

---

## Performing Calculations

You can write statements that perform simple calculations using mathematical operators. For example, you can use a multiplication operator to multiply the values displayed in two text boxes in your Web site. VBScript supports the following mathematical operators:

| Operator | Description |
|---|---|
| + | addition |
| – | subtraction |
| – | negation (a negative number) |
| * | multiplication |
| / | division |
| ^ | exponent |

In the following example, the multiplication operator is used to calculate the maximum luggage weight of the passengers at 100 pounds each:

```
'Retrieves the number of passengers from the edit box using the GetROProperty method

passenger = Browser ("Mercury_Tours").Page ("Find_Flights").
    WebEdit("numPassengers").GetROProperty("value")

'Multiplies the number of passengers by 100

weight = passenger * 100

'Inserts the maximum weight into a message box.

msgbox("The maximum weight for the party is "& weight &"pounds.")
```

## For...Next Statement

A For...Next loop instructs QuickTest to perform one or more statements a specified number of times. It has the following syntax:

**For** *counter* = *start* **to** *end* **[Step** *step***]**
    *statement*
**Next**

| Item | Description |
|---|---|
| *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.<br>**Default** = **1**.<br>**Optional**. |
| *statement* | A statement, or series of statements, to be performed during the loop. |

In the following example, QuickTest calculates the factorial value of the number of passengers using the For statement:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numPassengers").GetROProperty("value")
total = 1
For i=1 To passengers
    total = total * i
Next
MsgBox "!" & passengers & "=" & total
```

## For...Each Statement

A For...Each loop instructs QuickTest to perform one or more statements for each element in an array or an object collection. It has the following syntax:

**For Each** *item* **In** *array*
    *statement*
**Next**

| Item | Description |
|---|---|
| *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
```

## Do...Loop Statement

The Do...Loop statement instructs QuickTest to perform a statement or series of statements while a condition is true or until a condition becomes true. It has the following syntax:

**Do [{while} {until}** c*ondition*]
    *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, QuickTest calculates the factorial value of the number of passengers using the Do...Loop:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numPassengers").GetROProperty("value")
total = 1
i = 1
Do while i <= passengers
     total = total * i
     i = i + 1
Loop
MsgBox "!" & passengers & "=" & total
```

## While...Wend Statement

A While...Wend statement instructs QuickTest to perform a statement or series of statements while a condition is true. It has the following syntax:

**While** *condition*
    *statement*
**Wend**

| Item | Description |
|---|---|
| *condition* | A condition to be fulfilled. |
| *statement* | A statement or series of statements to be executed during the loop. |

In the following example, QuickTest performs a loop using the While statement while the number of passengers is fewer than ten. Within each loop, QuickTest increments the number of passengers by one:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
While passengers < 10
    passengers = passengers + 1
Wend

msgbox("The number of passengers in the party is " & passengers)
```

## If...Then...Else Statement

The If...Then...Else statement instructs QuickTest to perform a statement or a series of statements based on specified conditions. If a condition is not fulfilled, the next Elseif condition or Else statement is examined. It has the following syntax:

**If** *condition* **Then**
    *statement*
**ElseIf** *condition*2 **Then**
    *statement*
**Else**
    *statement*
**End If**

| Item | Description |
|---|---|
| *condition* | Condition to be fulfilled. |
| *statement* | Statement to be perform. |

In the following example, if the number of passengers is fewer than four, QuickTest closes the browser:

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
If (passengers < 4) Then
    Browser("Mercury Tours").Close
Else
    Browser("Mercury Tours").Page("Find Flights").Image("continue").Click 69,5
End If
```

The following example uses If, ElseIf, and Else statements to check whether a value is equal to 1, 2, or a different value:

```
value = 2
If value = 1 Then
  msgbox "one"
ElseIf value = 2 Then
  msgbox "two"
Else
  msgbox "not one or two"
End If
```

## Retrieving and Setting Test Object Property Values

Test object properties are the set of properties defined by QuickTest for each object. You can set and retrieve a test object's property values, and you can retrieve the values of test object properties from a run-time object.

When you run your test or component, QuickTest creates a temporary version of the test object that is stored in the test object repository. You can use the GetTOProperty, GetTOProperties, and SetTOProperty methods in your function library to set and retrieve the test object property values of the test object.

The GetTOProperty and GetTOProperties methods enable you to retrieve a specific property value or all the properties and values that QuickTest uses to identify an object.

The SetTOProperty method enables you to modify a property value that QuickTest uses to identify an object.

---

**Note:** Because QuickTest refers to the temporary version of the test object during the run session, any changes you make using the SetTOProperty method apply only during the course of the run session, and do not affect the values stored in the test object repository.

---

For example, the following statements would set the **Submit** button's name value to my button, and then retrieve the value my button to the **ButtonName** variable:

```
Browser("QA Home Page").Page("QA Home Page").
    WebButton("Submit").SetTOProperty "Name", "my button"

ButtonName=Browser("QA Home Page").Page("QA Home Page").
    WebButton("Submit").GetTOProperty("Name")
```

You use the GetROProperty method to retrieve the current value of a test object 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 test object properties of objects in your application, you can view them using the Object Spy. For information on the Object Spy, see Chapter 3, "Understanding the Test Object Model."

---

For a list and description of test object properties supported by each object, and for more information on the GetROProperty, GetTOProperty, GetTOProperties, and SetTOProperty methods, see the *HP QuickTest Professional Object Model Reference*.

# Accessing Run-Time Object Properties and Methods

If the test object methods and properties available for a particular test object do not provide the functionality you need, you can access the native methods and properties of any run-time object in your application using the Object property.

You can use the statement completion feature with object properties to view a list of the available native methods and properties of an object. For more information on the statement completion option, see "Generating Statements in a Function Library" on page 312.

---

**Tip:** If the object is a Web object, you can also reference its native properties in programmatic descriptions using the attribute/property notation. For more information, see "Accessing User-Defined Properties of Web Objects" on page 355.

---

## Retrieving Run-Time Object Properties

You can use the Object property to access the native properties of any run-time object. For example, you can retrieve the current value of the ActiveX calendar's internal Day property as follows:

```
Dim MyDay
Set MyDay=
Browser("index").Page("Untitled").ActiveX("MSCAL.Calendar.7").Object.Day
```

For more information on the Object property, see the *HP QuickTest Professional Object Model Reference*.

## Activating Run-Time Object Methods

You can use the Object property to activate the internal methods of any run-time object. For example, you can activate the native focus method of the edit box as follows:

```
Dim MyWebEdit
Set MyWebEdit=Browser("Mercury Tours").Page("Mercury Tours").
    WebEdit("username").Object
MyWebEdit.focus
```

For more information on the Object property, see the *HP QuickTest Professional Object Model Reference*.

## Accessing User-Defined Properties of Web Objects

You can use the attribute/<property name> notation to access native properties of Web objects and use these properties to identify such objects with programmatic descriptions.

For example, suppose a Web page has the same company logo image in two places on the page:

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

You could identify the image that you want to click using a programmatic description by including the user-defined property LogoID in the description as follows:

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

For more information on programmatic descriptions, see "Using Programmatic Descriptions" on page 332.

## Running DOS Commands

You can run standard DOS commands in your QuickTest 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 information, see the Microsoft VBScript Language Reference (choose **Help** > **QuickTest Professional Help** > **VBScript Reference** > **VBScript**).

## Enhancing Your Tests and Function Libraries Using the Windows API

Using the Windows API, you can extend testing abilities and add usability and flexibility to your function libraries. The Windows operating system provides a large number of functions to help you control and manage Windows operations. You can use these functions to obtain additional functionality.

The Windows API is documented in the Microsoft MSDN Web site, which can be found at: http://msdn2.microsoft.com/en-us/library/Aa383750

A reference to specific API functions can be found at: http://msdn2.microsoft.com/en-us/library/Aa383749

**To use Windows API functions:**

1. In MSDN, locate the function you want to use in your function library.
2. Read its documentation and understand all required parameters and return values.

**3** Note the location of the API function. API functions are located inside Windows DLLs. The name of the DLL in which the requested function is located is usually identical to the Import Library section in the function's documentation. For example, if the documentation refers to **User32.lib**, the function is located in a DLL named **User32.dll**, typically located in your System32 library.

**4** Use the QuickTest Extern object to declare an external function. For more information, see the *HP QuickTest Professional Object Model Reference*.

The following example declares a call to a function called **GetForegroundWindow**, located in **user32.dll**:

```
extern.declare micHwnd, "GetForegroundWindow", "user32.dll",
"GetForegroundWindow"
```

**5** Call the declared function, passing any required arguments, for example, `hwnd = extern.GetForegroundWindow()`.

In this example, the foreground window's handle is retrieved. You can enhance your function library if the foreground window is not in the object repository or cannot be determined beforehand (for example, a window with a dynamic title). You may want to use this handle as part of a programmatic description of the window, for example:

```
Window("HWND:="&hWnd).Close
```

In some cases, you may have to use predefined constant values as function arguments. Since these constants are not defined in the context of your function, you need to find their numerical value to pass them to the called function. The numerical values of these constants are usually declared in the function's header file. A reference to header files can also be found in each function's documentation under the Header section. If you have Microsoft Visual Studio installed on your computer, you can typically find header files under `X:\Program Files\Microsoft Visual Studio\VC98\Include`.

For example, the GetWindow API function expects to receive a numerical value that represents the relationship between the specified window and the window whose handle is to be retrieved. In the MSDN documentation, you can find the constants: GW_CHILD, GW_ENABLEDPOPUP, GW_HWNDFIRST, GW_HWNDLAST, GW_HWNDNEXT, GW_HWNDPREV and GW_HWNDPREV. If you open the **WINUSER.H** file, mentioned in the GetWindow documentation, you will find the following flag values:

```
/*
 * GetWindow() Constants
 */
#define GW_HWNDFIRST0
#define GW_HWNDLAST 1
#define GW_HWNDNEXT2
#define GW_HWNDPREV 3
#define GW_OWNER 4
#define GW_CHILD 5
#define GW_ENABLEDPOPUP 6
#define GW_MAX 6
```

## Example

The following example retrieves a specific menu item's value in the Notepad application.

```
' Constant Values:
const MF_BYPOSITION = 1024
' API Functions Declarations
Extern.Declare micHwnd,"GetMenu","user32.dll","GetMenu",micHwnd
Extern.Declare
micInteger,"GetMenuItemCount","user32.dll","GetMenuItemCount",micHwnd
Extern.Declare
micHwnd,"GetSubMenu","user32.dll","GetSubMenu",micHwnd,micInteger
Extern.Declare
micInteger,"GetMenuString","user32.dll","GetMenuString",micHwnd,micInteger,
    micString+micByRef,micInteger,micInteger
' Notepad.exe
hwin = Window("Notepad").GetROProperty ("hwnd")' Get Window's handle
MsgBox hwin
' Use API Functions
men_hwnd = Extern.GetMenu(hwin)' Get window's main menu's handle
MsgBox men_hwnd
item_cnt = Extern.GetMenuItemCount(men_hwnd)
MsgBox item_cnt
hSubm = Extern.GetSubMenu(men_hwnd,0)
MsgBox hSubm
rc = Extern.GetMenuString(hSubm,0,value,64 ,MF_BYPOSITION)
MsgBox value
```

# Choosing Which Steps to Report During the Run Session

You can use the Report.Filter method to determine which steps or types of steps are included in the Test 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.

The following report modes are available:

| Mode | Description |
|---|---|
| **0** or **rfEnableAll** | All events are displayed in the Test Results. **Default.** |
| **1** or **rfEnableErrorsAndWarnings** | Only events with a warning or fail status are displayed in the Test Results. |
| **2** or **rfEnableErrorsOnly** | Only events with a fail status are displayed in the Test Results. |
| **3** or **rfDisableAll** | No events are displayed in the Test Results. |

➤ To disable reporting of subsequent steps, enter the following statement:

```
Reporter.Filter = rfDisableAll
```

➤ To re-enable reporting of subsequent steps, enter:

```
Reporter.Filter = rfEnableAll
```

➤ To instruct QuickTest to include only subsequent failed steps in the Test Results, enter:

```
Reporter.Filter = rfEnableErrorsOnly
```

➤ To instruct QuickTest to include only subsequent failed or warning steps in the Test Results, enter:

```
Reporter.Filter = rfEnableErrorsAndWarnings
```

➤ To retrieve the current report mode, enter:

```
MyVar=Reporter.Filter
```

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

# 10

# Customizing Function Library Windows

You can customize the way functions are displayed in the function library windows. Any changes you make are applied globally to all function library windows.

**This chapter includes:**



## About Customizing Function Library Windows

QuickTest includes a powerful and customizable editor that enables you to modify many aspects of function library windows.

The Editor Options dialog box enables you to change the way function libraries are displayed in function library windows. You can also change the font style and size of text in your function libraries, and change the color of different elements, including comments, strings, QuickTest reserved words, operators, and numbers. For example, you can display all text strings in red.

QuickTest includes a list of default keyboard shortcuts that enable you to move the cursor, delete characters, and cut, copy, and paste information to and from the Clipboard. You can replace these shortcuts with shortcuts you prefer. For example, you could change the **Line start** command from the default HOME to ALT + HOME.

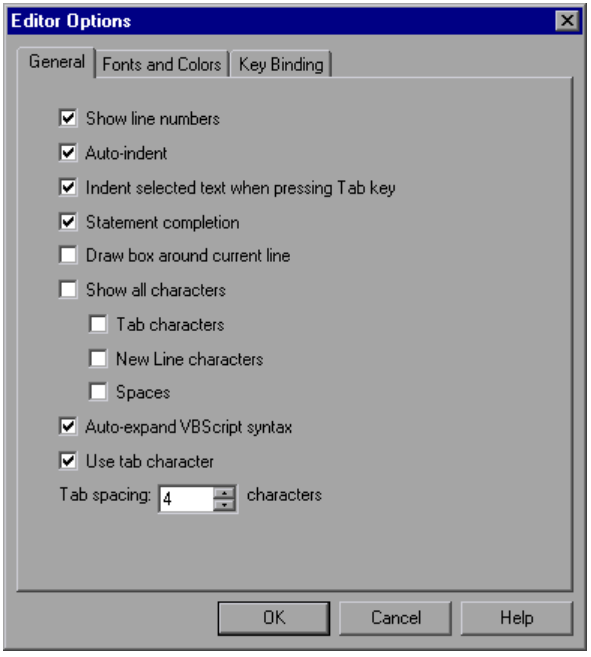
You can also modify the way your function library is printed using options in the Print dialog box. For more information, see “Printing a Function Library” on page 384. For more information on working with function libraries, see Chapter 11, “Working with User-Defined Functions and Function Libraries.”

## Customizing Editor Behavior

You can customize how function libraries are displayed in function library windows. For example, you can show or hide character symbols, and choose to display line numbers. For more information on working with function libraries, see Chapter 11, “Working with User-Defined Functions and Function Libraries.”

**To customize editor behavior:**

1. When a function library window is active, choose **Tools** > **View Options**. The Editor Options dialog box opens.
2. Click the **General** tab.



**3** Choose from the following options:

| Options | Description |
|---|---|
| **Show line numbers** | Displays a line number to the left of each line in the function. |
| **Auto-indent** | Causes lines following an indented line to automatically begin at the same point as the previous line. You can click the HOME key on your keyboard to move the cursor back to the left margin. |
| **Indent selected text when pressing Tab key** | Pressing the TAB key indents the selected text. When this option is not enabled, pressing the Tab key replaces the selected text with a single Tab character. |
| **Statement completion** | When this option is selected, if you type:<br>➤ **a dot after a test object.** QuickTest displays a list of available test objects and methods that you can add after the object you typed.<br>➤ **an open parenthesis after an object.** QuickTest displays a list of all test objects of this type in the object repository. If there is only one object of this type in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis.<br>➤ **a method.** QuickTest displays the syntax for the method, including its specific mandatory and optional arguments.<br>➤ **the Object property.** If the object data is currently available in the open application, QuickTest displays native methods and properties of any run-time object in your application. |
| **Draw box around current line** | Displays a box around the line of the test in which the cursor is currently located. |

| Options | Description |
|---|---|
| **Show all characters** | Displays all TAB, NEW LINE, and SPACE character symbols. You can also select to display only some of these characters by selecting or clearing the relevant check boxes. |
| **Auto-expand VBScript syntax** | Automatically recognizes the first two characters of keywords and adds the relevant VBScript syntax or blocks to the script, when you type the relevant keyword.<br>For example, if you enter the letters if and then enter a space at the beginning of a line in the Expert View, QuickTest automatically enters:<br>If Then<br>End If |
| **Use tab character** | Inserts a TAB character when the TAB key on the keyboard is used. When this option is not enabled, the specified number of space characters is inserted when you press the TAB key. |

**4** Click **OK** to apply the changes and close the dialog box.

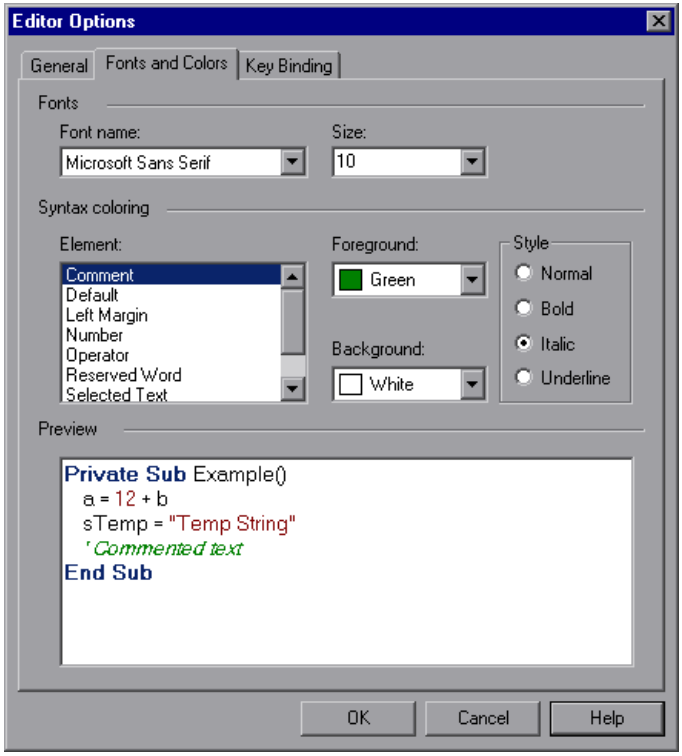
# Customizing Element Appearance

QuickTest function libraries contain many different elements, such as comments, strings, QuickTest and VBScript reserved words, operators, and numbers. Each element of QuickTest function libraries can be displayed in a different color. You can also specify the font style and size to use for all elements. You can create your own personalized color scheme for each element. For example, all comments could be displayed as blue letters on a yellow background.

**To set font and color preferences for elements:**

1 When a function library window is active, choose **Tools** > **View Options**. The Editor Options dialog box opens.

2 Click the **Fonts and Colors** tab.



3 In the **Fonts** area, select the **Font name** and **Size** that you want to use to display all elements. By default, the editor uses the Microsoft Sans Serif font, which is a Unicode font.

---

**Note:** When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your function library may not be correctly displayed in the function library windows. However, the function library will still run in the same way, regardless of the font you choose. If you are working in an environment that is not Unicode-compatible, you may prefer to choose a fixed-width font, such as Courier, to ensure better character alignment.

---

4 Select an element from the **Element** list.

5 Choose a foreground color and a background color.

6 Choose a font style for the element (**Normal**, **Bold**, **Italic**, or **Underline**). An example of your change is displayed in the **Preview** pane at the bottom of the dialog box.

7 Repeat steps4 to6 for each element you want to modify.

8 Click **OK** to apply the changes and close the dialog box.

# Personalizing Editing Commands

You can personalize the default keyboard shortcuts you use for editing. QuickTest includes keyboard shortcuts that let you move the cursor, delete characters, and cut, copy, or paste information to and from the Clipboard. You can replace these shortcuts with your preferred shortcuts. For example, you could change the **Line end** command from the default END to ALT + END.

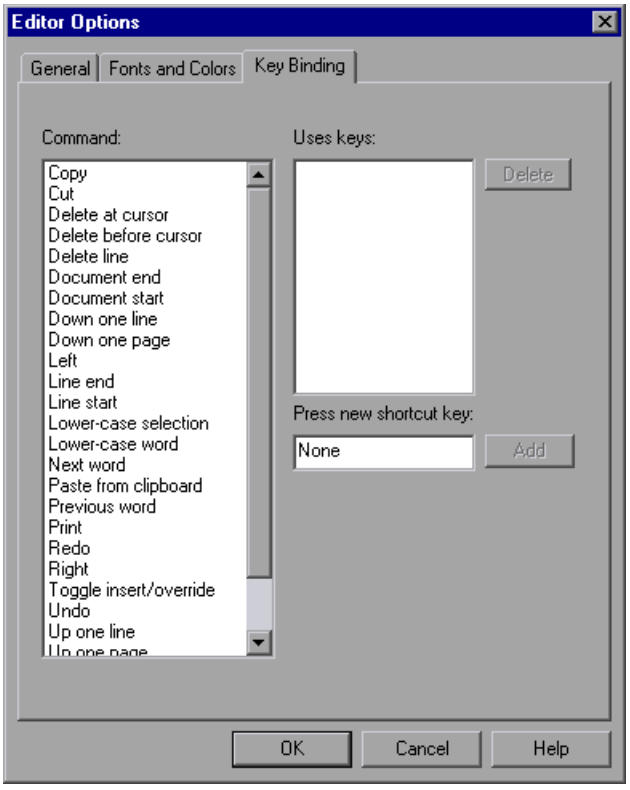
---

**Note:** The default QuickTest menu shortcut keys override any key bindings that you may define. For example, if you define the Paste command key binding to be CTRL+P, it will be overridden by the default QuickTest shortcut key for opening the Print dialog box (corresponding to the **File > Print** option). For a complete list of QuickTest menu shortcut keys, see "Performing QuickTest Commands" on page 71.

---

**To personalize editing commands:**

1. When a function library window is active, choose **Tools** > **View Options**. The Editor Options dialog box opens.
2. Click the **Key Binding** tab.




3. Select a command from the **Command** list.
4. Click in the **Press new shortcut key** box and then press the keys you want to use for the selected command. For example, press and hold the CTRL key while you press the number 4 key to enter CTRL+4.

**5** Click **Add**.

---

**Note:** If the key combination you specify is not supported, or if it is already defined for another command, a message to this effect is displayed below the shortcut key box.

---

**6** Repeat steps 3 - 5 for any additional commands.

**7** If you want to delete a key sequence from the list, select the command in the **Command** list, then highlight the keys in the **Uses keys** list, and click **Delete**.

**8** Click **OK** to apply the changes and close the dialog box.

# 11

# Working with User-Defined Functions and Function Libraries

In addition to the test objects, methods, and built-in functions supported by the QuickTest Test Object Model, you can define your own function libraries containing VBScript functions, subroutines, modules, and so forth, and then use their functions as operations in your component.

---

**Note:** The terms function, method, and operation are used interchangeably in this chapter. This is because functions and methods are known as operations in the Business Component Keyword View, whereas in QuickTest, the terms function and method are used.

---

**This chapter includes:**

# About Working with User-Defined Functions and Function Libraries

You can create user-defined functions to provide additional functionality for your components. 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 components are easier to design, read, and maintain. You or a Subject Matter Expert can then call user-defined functions from a component by inserting the relevant keywords (or operations) into that component.

You can register a user-defined function as a method for a QuickTest test object. 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. For more information on registering user-defined functions, see "Using the Function Definition Generator" on page 390 and "Registering User-Defined Functions as Test Object Methods" on page 404.

---

**Note:** When you create a user-defined function, do not give it the same name as a built-in function (for example, GetLastError, MsgBox, or Print). Built-in functions take priority over user-defined functions, so if you call a user-defined function that has the same name as a built-in function, the built-in function is called instead. For a list of built-in functions, see the **Built-in functions** list in the Step Generator (**Insert > Step Generator**).

---

Using QuickTest, you can define and store your user-defined functions in a function library (saved as a **.qfl** file, by default). A function library is a Visual Basic script containing VBscript functions, subroutines, modules, and so forth. You can also use QuickTest to modify and debug any existing function libraries (such as **.vbs** or **.txt** files). For information on using VBScript, see "Handling VBScript Syntax Errors" on page 330.

When you store a function in a function library and associate the function library with an application area, any component associated with that application area can call the public functions in that function library. For more information, see "Working with Associated Function Libraries" on page 387. Functions that are stored in an associated function library can be accessed from the Step Generator (for function libraries) and the **Operation** column in the Keyword View.

You can also define private functions and store them in a function library. Private functions are functions that can be called only by other functions within the same function library. This is useful if you to reuse segments of code in your public functions.

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, or set the function as the default method for the test object. For more information, see "Using the Function Definition Generator" on page 390.

## Managing Function Libraries

You can create function libraries in QuickTest and call their functions from your component after you associate the function library with the component's application area. A function library is a separate QuickTest document containing VBscript functions, subroutines, modules, and so forth. Each function library opens in a separate window, enabling you to open and work on one or several function libraries at the same time. After you finish editing a function library, you can close it, leaving your QuickTest session open. You can also close all open function libraries simultaneously.

By implementing user-defined functions in function libraries and associating them with your component via the application area, you enable other users, such as Subject Matter Experts, to choose functions that perform complex operations, such as adding if/then statements and loops to component steps—without needing any programming knowledge. In addition, you save time and resources by implementing and using reusable functions.

QuickTest provides tools that enable you to edit and debug any function library, even if it was created using an external editor. For example, QuickTest can check the syntax of your functions, and the function library window provides the same editing features that are available in the Expert View. For more information on the options available in the Expert View, see the *HP QuickTest Professional User's Guide*.

## Creating a Function Library

You can create a new function library at any time.

**To create a new function library in QuickTest:**

Perform one of the following:

➤ Choose **File** > **New** > **Function Library**

➤ Click the **New** button down arrow and choose **Function Library**

A new function library opens.

You can now add content to your function library and/or save it. When you add content to your function library, QuickTest applies the same formatting it applies to content in the Expert View. You can modify the formatting, if needed. For more information, see "Customizing Function Library Windows" on page 363.

## Opening a Function Library

In QuickTest, you can open any function library that is saved in the file system or your Quality Center project—even if another document is already open in QuickTest. You can only open a function library if you have read or read-write permissions for the file.

---

**Note:** To enable a component or application area to use the functions defined in a function library, the function library must be saved in your Quality Center project and be associated with the application area. For more information, see "Managing Function Libraries" on page 426.

---

You can 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.

---

**Tip:** You can also navigate directly from a function in your document to its function definition in another function library. For more information, see "Navigating to a Specific Function in a Function Library" on page 382.

---

**To open an existing function library:**

Perform one of the following:

- Choose **File** > **Open** > **Function Library**
- Click the **Open** button down arrow and choose **Function Library**

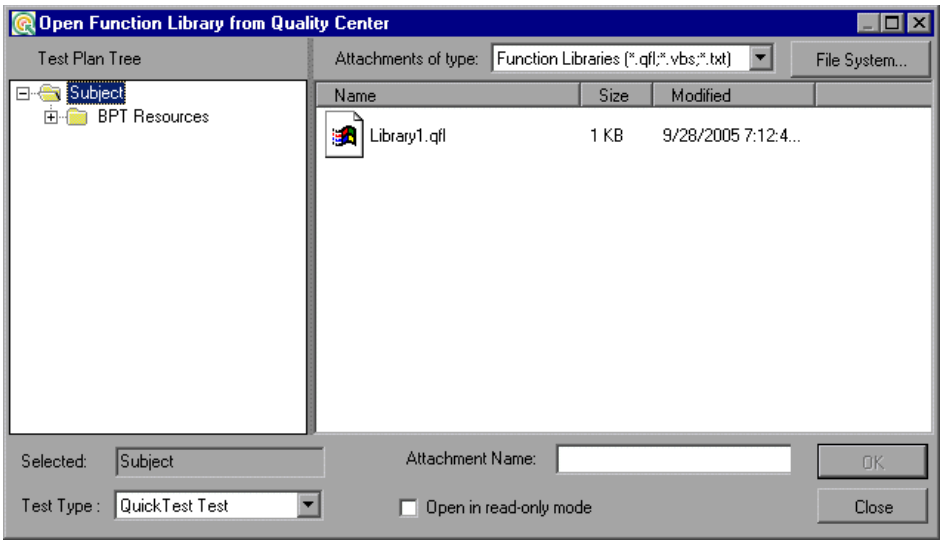
---

**Tips:**

- If the function library was recently created or opened, you can choose it from the recent files list in the **File** menu.
- If the function library is associated with the open component or application area, you can choose it from **Resources** > **Associated Function Libraries**.

---

The Open Function Library from Quality Center dialog box opens.




---

**Tip:** You can open the function library in read-only mode by selecting the **Open in read-only mode** check box.

---

Browse to and select a function library, and click **Open**. QuickTest opens the specified function library in a new window. You can now view and modify its content. For more information, see “Editing a Function Library” on page 382 and “Debugging a Function Library” on page 384.

## Saving a Function Library

After you create or edit a function library in QuickTest, you can save it to your Quality Center project.

---

**Tips:**

- When you modify a function library, an asterisk (*) is displayed in the title bar until the function library is saved.
- To save all open documents, choose **File** > **Save All**. QuickTest prompts you to specify a location in which to save any new files that have not yet been saved.
- To save multiple documents, choose **Window** > **Windows**. In the Window dialog box, select the documents you want to save and click the **Save** button. QuickTest prompts you for the save location for any new files that have not yet been saved.
- You can also choose **File** > **Save As** to save the active function library under a different name or using a different path.

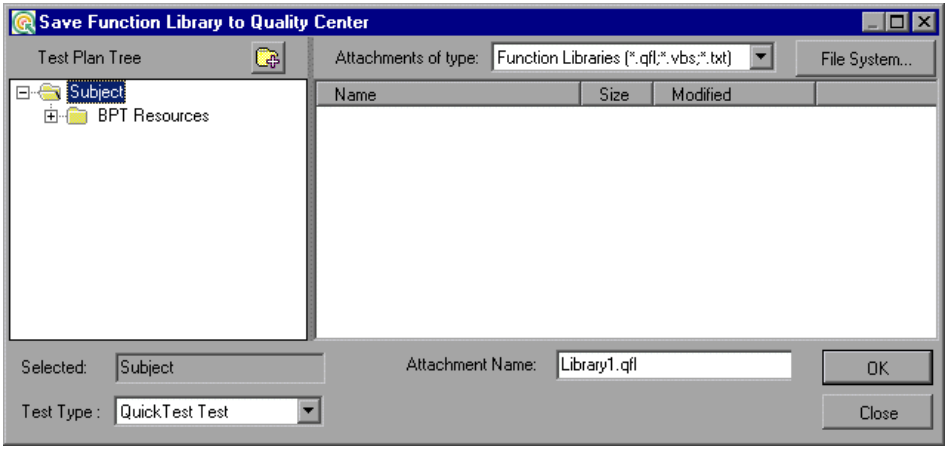
---

**To save a function library:**

1. Make sure that the function library you want to save is the active document. (You can click the function library's tab to bring it into focus.)
2. Perform one of the following:

- Click the **Save** button.
- Choose **File** > **Save**.
- Right-click the function library document's tab and choose **Save**.

If the function library was previously saved, QuickTest saves it with your changes. Otherwise, if this is the first time you are saving this function library, the Save Function Library to Quality Center dialog box opens.




**3** Save the function library to your Quality Center project.

**4** In the Test Plan Tree box, choose the folder in which you want to save the function library.

---

**Note:** You must save the function library in your Quality Center project (not the file system).

---

**5** In the **Attachment name** box, type a name for the function library.

**6** Click **OK**.

QuickTest saves the function library with a **.qfl** extension (unless you specify a different extension, such as **.vbs** or **.txt**, or remove the extension altogether), and displays the function library name in the title bar.

## Navigating Between Open QuickTest Documents

You can open multiple function libraries while a component or application area is open, and you can navigate between all of your open documents.

**To navigate between open QuickTest documents:**

Perform one of the following:

➤ Click the tab for the required document in the Document pane

---

**Tip:** If not all tabs are displayed due to lack of space, use the left and right scroll arrows in the Document pane to display the required document's tab.

---

➤ Press CTRL+TAB on your keyboard to scroll between open documents

➤ Choose the required document from the **Window** menu

➤ Choose **Window > Windows**, select the required document in the Windows dialog box, and click the **Activate** button

---

**Note:** You can also choose **Resources > Associated Function Libraries** and choose the required function library from the list. This also opens closed function libraries that are associated with your component or application area.

---

## Navigating to a Specific Function in a Function Library

After you insert a call to a function, you can navigate directly to its definition in the source document. The function definition can be located either in the same function library or in another function library that is associated with your component (via its application area). If the document containing the function definition is already open, QuickTest activates the window (brings the window into focus). If the document is closed, QuickTest opens it.

**To navigate to a function's definition:**

1 In the function library, click in the step containing the relevant function.

2 Perform one of the following:

- Choose **Edit** > **Advanced** > **Go to Function Definition**.
- Right-click the step and choose **Go to Function Definition** from the context menu.

QuickTest activates the relevant document (if the function definition is located in another function library) and positions the cursor at the beginning of the function definition.

## Editing a Function Library

You can edit a function library at any time using the QuickTest editing features that are available in the Expert View.

You can drag and drop a function (or part of it) from one document to another. (To do so, you must first separate the tabbed documents into separate document panes by clicking the **Restore Down** button (located below the QuickTest window's **Restore Down** / **Maximize** button).)

You can add steps to your function library manually or using the Step Generator. The Step Generator enables you to add steps that contain **reserved objects** (the objects that QuickTest supplies for enhancement purposes, such as utility objects), VBScript functions (such as MsgBox), utility statements (such as Wait), and user-defined functions that are defined in the same function library. IntelliSense is available for all functions defined in your component or for public functions defined in associated function libraries.

---

**Note:** In function libraries, IntelliSense does not enable you to view test object names or collections because function libraries are not connected to object repositories.

---

You can instruct QuickTest to check syntax by clicking the **Check Syntax** button, or by choosing **Tools** > **Check Syntax**.

---

**Tip:** For information on using VBScript, see "Understanding Basic VBScript Syntax" on page 325.

---

## Editing a Read-Only Function Library

If you open a function library in read-only mode and then decide to modify it, you can convert the function library to an editable file—as long as the function library is not locked by another user. For more information on the options available when opening a function library, see "Opening a Function Library" on page 376.

---

**Note:** During a debug session, all documents (such as components and function libraries) are read-only. To edit a document during a debug session, you must first stop the debug session.

---

**To edit a read-only function library:**

Choose **File** > **Enable Editing** or click the **Enable Editing** button. You can now edit the function library.

## Debugging a Function Library

Before you can debug a function library, you must first associate it with a component (via its application area) and then insert a call to at least one of its functions. For example, you can use the Debug Viewer to view, set, or modify the current value of objects or variables in your function library. You can step into functions (including user-defined functions), set breakpoints, stop at breakpoints, view expressions, and so forth. You can begin debugging from a specific step, or you can instruct QuickTest to pause at a specific step. For more information, see "Debugging Components and Function Libraries" on page 681.

---

**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.

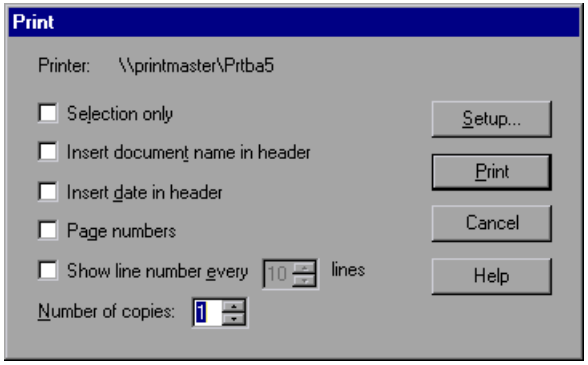
---

## Printing a Function Library

You can print a function library at any time. You can also include additional information in the printout.

**To print from the function library:**

**1** Click the **Print** button or choose **File** > **Print**. The Print dialog box opens.




**2** Specify the print options that you want to use:

- **Printer.** Displays the printer to which the print job will be sent. You can change the printer by clicking the **Setup** button.
- **Selection only.** Prints only the text that is currently selected (highlighted) in the function library.
- **Insert document name in header.** Includes the name of the function library at the top of the printout.
- **Insert date in header.** Includes today's date at the top of the printout. The date format is taken from your Windows regional settings.
- **Page numbers.** Includes page numbers on the bottom of the printout (for example, page 1 of 3).
- **Show line numbers every __ lines.** Displays line numbers to the left of the script lines, as specified.
- **Number of copies.** Specifies the number of times to print the document.

**3** If you want to print to a different printer or change your printer preferences, click **Setup** to display the Print Setup dialog box.

**4** Click **Print** to print according to your selections.

## Closing a Function Library

You can close an individual function library, or if you have several function libraries open, you can close some or all of them simultaneously. If any of the function libraries are not saved, QuickTest prompts you to save them.

**To close an individual function library:**

Perform one of the following:

- Make sure that the function library you want to save is the active document—you can click the function library's tab to bring it into focus—and choose **File** > **Close**.
- Right-click the function library document's tab and choose **Close**.
-  Click the **Close** button in the top right corner of the function library window.
- Choose **Window** > **Windows**. In the Windows dialog box, select the function library to close if it is not already selected, and click the **Close Window(s)** button.

**To close several function libraries:**

Choose **Window** > **Windows**. In the Windows dialog box, select the function libraries you want to close and click the **Close Window(s)** button.

**To close all open function libraries:**

Choose **File** > **Close All Function Libraries**, or **Window** > **Close All Function Libraries**.

# Working with Associated Function Libraries

In QuickTest, you can create function libraries containing functions, subroutines, modules, and so forth, and then associate the files with your application area. This enables you or a Subject Matter Expert to insert a call to a public function or subroutine in the associated function library from any component associated with that application area. (Public functions stored in function libraries can be called from any associated component (via its application area), whereas private functions can be called only from within the same function library.)

If a component can no longer access a function that was used in a step (for example, if the function was deleted from the associated function library), the  icon is displayed adjacent to the step in the Keyword View. When you run the component or business process test, an error will occur when it reaches the step using the nonexistent function.

---

**Note:** Any text file written in standard VBScript syntax can be used as a function library.

---

You can edit the list of associated function libraries for an existing application area in the Function Libraries pane in an application area (**Application Area** > **Function Libraries** sidebar button). For more information, see "Managing Function Libraries" on page 426.

## Working with Associated Function Libraries in Quality Center

When working with Quality Center and associated function libraries, you must save the associated function library as an attachment in your Quality Center project before you specify the associated file in the Function Libraries pane of the application area. You can add a new or existing function library to your Quality Center project.

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 Quality Center project and associated with an application area may affect its associated components. When making changes to a function library that is stored in your Quality Center project and associated with an application area, consider the effect of the changes on the components that use this application area.

## Associating a Function Library with an Application Area

You can associate a function library with an open application area either from the Resources pane or from the currently active function library.

You can also associate function libraries with the currently open application area using the associated function libraries list. For more information, see "Modifying Function Library Associations" on page 389.

**To associate a function library with an application area using the Resources pane:**

1. In the Resources pane, right-click the **Associated Function Libraries** node in the tree, and select **Associate Function Library**. The Open Attachment dialog box opens.
2. Browse to and select the function library you want to associate.
3. Click **Open**. The function library is associated with the test, and is displayed as a function library link in the **Associated Function Libraries** node in the tree.

**To associate an open function library with an application area:**

1. Make sure that the application area with which you want to associate the function library is open in QuickTest.
2. Create or open a function library in QuickTest. (Before continuing to the next step, make sure that the function library you want to associate with the application area is the active document—you can click the function library's tab to bring it into focus.) For more information, see "Managing Function Libraries" on page 375.
3. Save the function library in your Quality Center project as an attachment. For more information, see "Saving a Function Library" on page 379.

**4** In QuickTest, choose **File** > **Associate Library** '<**Function Library**>' **with** '<**Application Area**>' or right-click in the in the function library and choose **Associate Library** '<**Function Library**>' **with** '<**Application Area**>'. QuickTest associates the function library with the open application area.

## Modifying Function Library Associations

You can modify the list of associated function libraries for an application area. You can add or remove function libraries from the list, and change their priorities.

**To modify function library associations in your application area:**

**1** In QuickTest, open your application area and click the **Function Libraries** button on the sidebar.

**2** In the associated function libraries list, click the **Add** button. QuickTest displays a browse button enabling you to browse to a function library in your Quality Center project.

**3** Select the function library you want to associate with your application area and click **OK**.

**To remove an associated function library:**

➤ In the Resources pane, right-click the function library and select **Remove Function Library**, or select the function library and press the DELETE key.

➤ In the list of associated function libraries in the Function Libraries pane, select the function library you want to remove and click the **Remove** button. You can also prioritize associated function libraries by using the **Up** and **Down** arrows.

For more information, see “Managing Function Libraries” on page 426.

# Using the Function Definition Generator

QuickTest provides a Function Definition Generator, which enables you to generate definitions for new user-defined functions and add header information to them. You can then register these functions to a test object, if needed. You fill in the required information and the Function Definition Generator creates the basic function definition for you. After you define the function definition, you insert the definition in your function library and associate it with your application area. Finally, you complete the function by adding its content (code).

If you register the function to a test object, it can be called by that test object, and is displayed in the list of available operations for that test object.

If you do not register the function to a test object, it becomes a global operation and is displayed in the list of operations in the **Operation** box in the Step Generator (for function libraries), in the **Operation** list when the **Operation** item is selected in the Keyword View, and when using IntelliSense. If you register a function, you can define it as the default operation that is displayed in the Keyword View when the test object to which it is registered is selected.

Finally, you can document your user-defined function by defining the tooltip that displays when the cursor is positioned over the operation in the Step Generator (for function libraries), in the Keyword View, and when using IntelliSense. You can also add a sentence that describes what the step that includes the user-defined function actually does. This sentence is then displayed in the **Documentation** column.

As you add information to the Function Definition Generator, the **Preview** area displays the emerging function definition. After you finish defining the function, you insert the definition in the active QuickTest document. The function will then be accessible to any associated component (via its application area). Finally, you add the content (code) of the function.

The following section provides an overview of the steps you perform when using the Function Definition Generator to create a function.

**To use the Function Definition Generator:**

1. Open the Function Definition Generator, as described in "Opening the Function Definition Generator" on page 392.
2. Define the function, as described in "Defining the Function Definition" on page 393.
3. Register the function to a test object, if needed, as described in "Registering a Function Using the Function Generator" on page 394.

   By default, functions that are not registered to a test object are automatically defined as global functions that can be called by selecting the **Functions** category in the Step Generator (for function libraries), the **Operation** item in the Keyword View, or when using IntelliSense. Note that if you register the function to a test object, you can also define the function (operation) as the default operation for that selected test object.
4. Add arguments to the function, as described in "Specifying Arguments for the Function" on page 398.
5. Document the function by adding header information to it, as described in "Documenting the Function" on page 399.
6. Preview the function before finalizing it, as described in "Previewing the Function" on page 401.
7. Generate another function definition, if needed, as described in "Generating Another User-Defined Function" on page 401.
8. Finalize each function by inserting it in your active document and adding content to it, as described in "Finalizing the User-Defined Function" on page 402.

---

**Note:** Each of the steps listed in this section assumes that you have performed the previous steps.

---

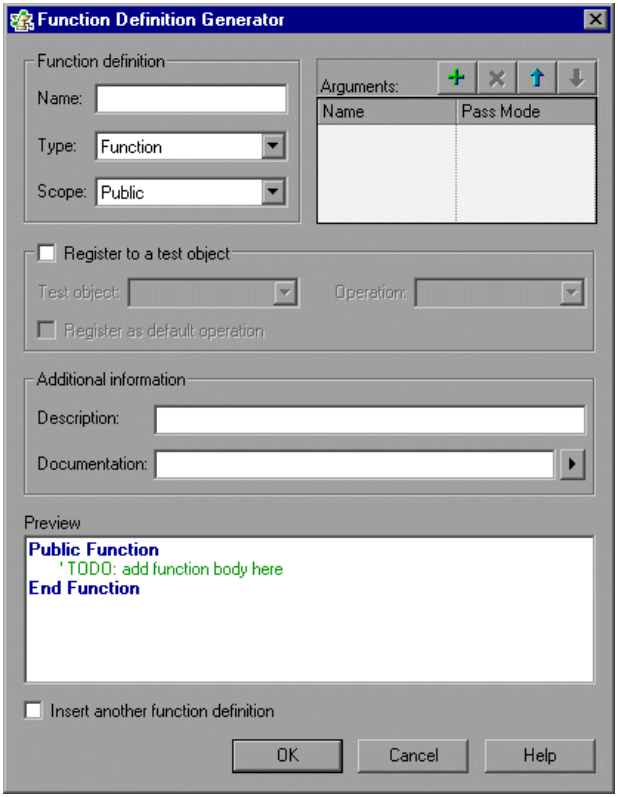
## Opening the Function Definition Generator

You open the Function Definition Generator from QuickTest.

**To open the Function Definition Generator:**

1 Make sure that the function library in which you want to insert the function definition is the active document. (You can click the function library'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.

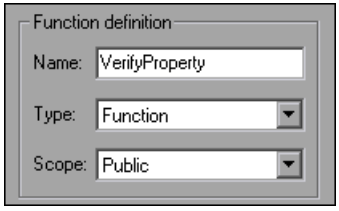
2 Choose **Insert** > **Function Definition Generator** or click the **Function Definition Generator** button. The Function Definition Generator opens.




After you open the Function Definition Generator, you can begin to define a new function.

## Defining the Function Definition

After you open the Function Definition Generator, you can begin defining a function.




For 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 component (as long as the function library is associated with its application area). (If you define it as private, the function can only be called from elsewhere in the same function library. Private functions cannot be registered to a test object.)

**To define a function:**

**1** In the **Name** box, enter 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 (for function libraries) or the Keyword View. Function names cannot contain non-English letters or characters. In addition, function names must begin with a letter and cannot contain spaces or any of the following characters:
! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ‘ : “ , / < > ?

---

**Note:** Do not give the user-defined function the same name as a built-in function (for example, GetLastError, MsgBox, or Print). Built-in functions take priority over user-defined functions, so if you call a user-defined function that has the same name as a built-in function, the built-in function is called instead. For a list of built-in functions, see the **Built-in functions** list in the Step Generator (**Insert** > **Step Generator**).

---

2 From the **Type** list, choose **Function** or **Sub**, according to whether you want to define a function or a subroutine.

3 From the **Scope** list, choose the scope of the function—either **Public** (to enable the function to be called by any component whose application area is associated with this function library), or **Private** (to enable the function to be called only from elsewhere in the same function library). By default, the scope is set to **Public**. (Only public functions can be registered to a test object.)

---

**Note:** If you create a user-defined function manually and do not define the scope as **Public** or **Private**, it will be treated as a public function, by default.

---

After you define a public function, you can register the function. Alternatively, if you defined a private function, or if you do not want to register the function, you can continue by specifying arguments for the function. For more information, see "Specifying Arguments for the Function" on page 398.

## Registering a Function Using the Function Generator

You can register a public function to a test object to enable the function (operation) to be performed on a test object. When you register a function to a test object, you can choose to override the functionality of an existing operation, or you can register the function as a new operation for the test object.

After you register a function to a test object, it is displayed as an operation in the Keyword View **Operation** list when that test object is selected from the **Item** list, as well as in IntelliSense and in the general **Operation** list in the Step Generator (for function libraries). When you register a function to a test object, it can only be called by that test object.

If you choose to register the function to a test object, the Function Definition Generator automatically adds the argument, **test_object**, as the first argument in the Arguments area in the top-right corner of the Function Definition Generator. The Function Definition Generator also automatically adds a RegisterUserFunc statement with the correct argument values immediately after your function definition.

When you register a function to a test object, you can optionally define it as the default operation for that test object. This instructs QuickTest to display the function in the **Operation** column, by default, when you or the Subject Matter Expert choose the associated test object in the **Item** list. When you define a function as the default function for a test object, the value **True** is specified as the fourth argument of the RegisterUserFunc statement.

If you do not register the function to a specific test object, the function is automatically defined as a global function. Global functions can be called by selecting the **Functions** category in the Step Generator (for function libraries), or the **Operation** item in the Keyword View. A list of global functions can be viewed alphabetically in the **Operation** box when the **Functions** category is selected in the Step Generator (for function libraries), in the **Operation** list when the **Operation** item is selected from the **Item** list in the Keyword View and when using IntelliSense.

QuickTest searches the function libraries in the order in which they are listed in the Resources tab. If QuickTest finds more than one function that matches the function name in a specific function library, it uses the last function it finds in that function library. If QuickTest finds two functions with the same name in two different function libraries, it uses the function from the function library that has the higher priority. To avoid confusion, it is recommended that you verify that within the resources associated with an application area, each function has a unique name.

---

**Tip:** If you choose not to register your function at this time, you can manually register it later by adding a RegisterUserFunc statement after your function as shown in the following example:
RegisterUserFunc "WebEdit", "MySet", "MySetFunc"

In this example, the MySet method (operation) is added to the WebEdit test object using the MySetFunc user-defined function. If you or the Subject Matter Expert choose the WebEdit test object from the **Item** list in the Keyword View, the MySet operation will then be displayed in the **Operation** list (together with other registered and out of the box operations for the WebEdit test object).

You can also register your function to other test objects by duplicating (copying and pasting) the RegisterUserFunc statement and modifying the argument values as needed when you save the function code in a function library.

To define this function as the default function, you define the value of the fourth argument of the RegisterUserFunc statement as **True**. For example:
RegisterUserFunc "WebEdit", "MySet", "MySetFunc", True

---

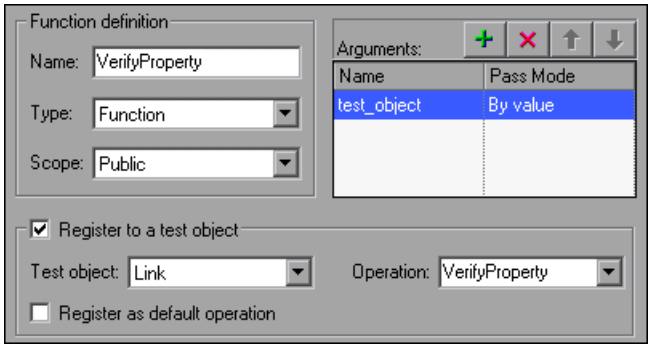
---

**Note:** A registered or global function can only be called from a component after it is added to a function library that is associated with the component's application area.

---

**To register the function to a test object:**

**1** Select the **Register to a test object** check box. The options in this area are enabled, and a new argument, test_object, is automatically added to the list of arguments in the **Arguments** area in the top-right corner of the Function Definition Generator. (The test_object argument receives the test object to which you want to register the function.)




---

**Note:** If you clear the **Register to a test object** check box, the default test_object argument is automatically removed from the **Arguments** area (unless you renamed it).

---

**2** Choose a **Test object** from the list of available objects. For example, for the sample VerifyProperty function, you might want to register it to the **Link** test object.

**3** Specify the **Operation** that you want to add or override for the test object.

- To define a new operation, enter a new operation name in the **Operation** box. For example, for the sample VerifyProperty function, you may want to define a new VerifyProperty operation.
- To override the standard functionality of an existing operation, choose an operation from the list of available operations in the **Operation** box.

**4** If you want the function to be displayed as the default operation in the **Operation** column when you or the Subject Matter Expert choose the associated item, select the **Register as default operation** check box.

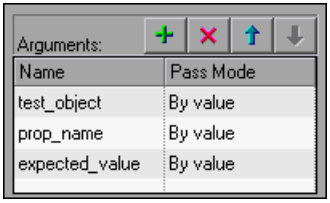
For example, if you were to define the VerifyProperty operation as the default operation for the Link test object, the value **True** would be defined as the fourth argument of the RegisterUserFunc statement, and the syntax would appear as follows:

```
RegisterUserFunc "Link", "VerifyProperty", "VerifyProperty", True
```

After you specify the test object registration information, you specify additional arguments for the function.

## Specifying Arguments for the Function

After you define the basic function definition and specify the test object registration information, if any, you can specify the function's arguments.




For example, if you choose to register the function to a test object, as we did the example described in "Registering a Function Using the Function Generator" on page 394, 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. You must define the required arguments for your function to run correctly.

You can list the arguments in any order. However, if you are registering the function to a test object, the first argument must always receive the test object.

**To define the arguments for the function:**

In the **Arguments** area, specify the arguments for the function. You can add as many arguments as needed. To ensure clarity, the name for each argument should indicate the value that needs to be entered.

- To add an argument, click  and enter a name for the argument. The argument name should clearly indicate the value that needs to be entered for the argument. Argument names may not contain non-English letters or characters. In addition, argument names must begin with a letter and cannot contain spaces or any of the following characters:
  ! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ‘ : “ “ , / < > ?

  By default, the **Pass Mode** is set as **By value**. This instructs QuickTest to pass the argument to the function by value. If you want to pass the argument by reference, choose **By reference** in the **Pass Mode** box.
- To remove an argument, select it and click  . The argument is removed from the Function Definition Generator.
- To set the order of the arguments, use the  and  arrows. The order of the arguments only affects the readability of the function code (except if you want to register the public function—in this case, the first argument must receive the test object).

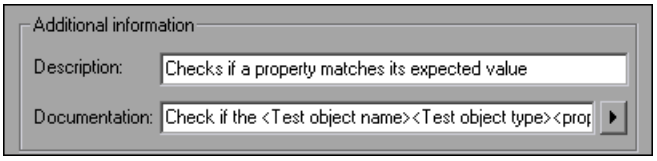
## Documenting the Function

The Function Definition Generator enables you to add header information to your user-defined function. You can add a description, which is displayed as a tooltip when the cursor is positioned over the operation. You and Subject Matter Experts can then use this tooltip to determine which operation to choose from the list of available operations. (It is advisable to keep the description text as brief and clear as possible.)

In addition, you can add documentation that specifies exactly what a step using your function does. You can include the test object name, test object type, and any argument values in the text. You can also add text manually, as needed. This text that you add here is displayed in the **Documentation** column. Therefore, the sentence must be clear and understandable.

For example, 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>.




After choosing values for the arguments in the Keyword View, the above documentation might appear as follows: Check if the "Management Software" link "text" value matches the expected value: "Business Technology Optimization (BTO) Software".

---

**Tip:** You can right-click on any column header in the Keyword View and select the **Documentation only** option to view or print a list of steps. This instructs QuickTest or Quality Center to display only the **Documentation** column (and any comments for business components). You can also choose **Edit** > **Copy Documentation to Clipboard** and then paste the documentation in any application. Therefore, the sentence displayed for the step in this column must also be clear enough to use for manual testing instructions.

---

**To document the function:**

1 In the **Description** box, enter the text to be displayed as a tooltip when the cursor is positioned over the function name in the **Operation** list in the Step Generator (for function libraries), in the **Operation** column in the Keyword View, and in IntelliSense.

For example, for the sample VerifyProperty function, you may want to enter: Checks whether a property value matches the actual value.

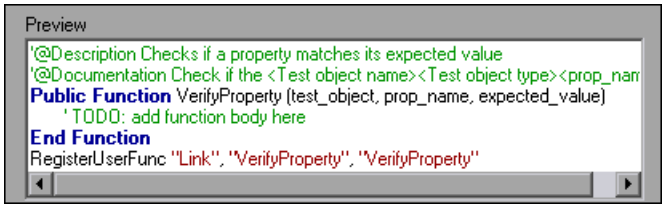
2 In the **Documentation** box, enter the text to be displayed in the **Documentation** column of the Keyword View. You can use arguments in the **Documentation** text by clicking ▸ and selecting the relevant argument.

If you selected the **Register to a test object** check box, clicking  also enables you to add the **Test object name** and/or **Test object type** items to the **Documentation** column from the displayed list. If you use these test object and argument items in the **Documentation** text, they are replaced dynamically by the relevant test object names and types or argument values.

## Previewing the Function

The **Preview** area displays the function code as you define it, in read-only format. You can review your function and make any changes, as needed, in the various areas of the Function Definition Generator.

For example, for the sample **VerifyProperty** function, the **Preview** area displays the following code.




After you review the code (before you insert it in the active document), you can choose either to generate another function definition or to finalize the code for the function you defined.

## Generating Another User-Defined Function

After you preview the code—before you insert the function in the active document—you can decide whether you want to generate an additional function definition.

---

**Note:** If you do not want to define an additional function, continue to the next section.

---

**To generate an additional user-defined function:**

1. Select the **Insert another function definition** check box and click **Insert**. QuickTest inserts the function definition in the active document and clears the data from the Function Definition Generator. The Function Definition Generator remains open.
2. Define the new function beginning from "Defining the Function Definition" on page 393.

## Finalizing the User-Defined Function

After you preview the code, you insert it in the active document. If you insert it in a function library, any component associated with the function library (via its application area) can access the function.

After you insert the code in the required location, you can finalize the function. For example, for the VerifyProperty function, the following code would be inserted in your function library:

```
'@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"
```

---

**Tip:** The RegisterUserFunc statement (in the last line) registers the VerifyProperty function to the Link test object. If you want to register the function to more than one test object, you could copy this line and duplicate it for each test object, changing the argument values, as required.

---

To finalize the function, you add its content (replacing the TODO comment). For example, if you want the function to verify whether the expected value of a property matches the actual property value of a specific test object, you might add the following to the body of the function:

```
Dim actual_value
    ' Get the actual property value
    actual_value = obj.GetROProperty(prop_name)
    ' Compare the actual value to the expected value
    If actual_value = expected_value Then
        Reporter.ReportEvent micPass, "VerifyProperty Succeeded", "The " &
prop_name & " expected value: " & expected_value & " matches the actual
value"
        VerifyProperty = True
    Else
        Reporter.ReportEvent micFail, "VerifyProperty Failed", "The " &
prop_name & " expected value: " & expected_value & " does not match the
actual value: " & actual_value
        VerifyProperty = False
    End If
```

**To finalize the user-defined function:**

1. Click **OK**. QuickTest inserts the function definition in the active document and closes the Function Definition Generator.
2. In your function library, add content to the function code, as required, by replacing the TODO line.

---

**Tip:** To display the function in the test results tree (Test Results window) after a run session, add a Reporter.ReportEvent statement to the function code (as shown in the example above).

Note that if your user-defined function uses a default test object method, this step will appear in the Test Results window 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 component or business process test status, if required.

---

**3** Associate the function library with an application area to enable access to the user-defined functions. You also need to check its syntax to ensure that components associated with that application area will have access to the functions, and that you and the Subject Matter Expert will be able to see and use the functions. For more information, see "Working with Associated Function Libraries" on page 387.

## Registering User-Defined Functions as Test Object Methods

In addition to using the QuickTest Function Definition Generator to register a function, as described in "Registering a Function Using the Function Generator" on page 394, you can also use the RegisterUserFunc statement to add new methods to test objects or to change the behavior of an existing test object method during a run session.

When you register a function to a test object, you can define it as the default operation for that test object, if required. The default operation is displayed by default in the **Operation** column in the Keyword View when the test object to which it is registered is selected.

If you choose not to register a function to a test object, it becomes a global function. Global functions can be called by selecting the **Functions** category in the Step Generator (for function libraries), the **Operation** item in the Keyword View, or when using IntelliSense. You use the UnregisterUserFunc statement to disable new methods or to return existing methods to their original QuickTest behavior.

To register a method, you first define a function in an associated function library. You then enter a RegisterUserFunc statement at the end of the function that specifies the test object class, the function to use, and the method name that calls your function. You can register a new method for a test object class, or you can use an existing method name to (temporarily) override the existing functionality of the specified method.

Your registered method applies only to the function library in which you register it. In addition, QuickTest clears all function registrations at the beginning of each run session.

## Preparing the User-Defined Function

When you run a statement containing a registered method, it 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. Make sure that if the function overrides an existing method, it has the exact syntax of the method it is replacing. This means that its first argument is the test object and the rest of the arguments match all the original method arguments.

---

**Tip:** You can use the **parent** test object property to retrieve the parent of the object represented by the first argument in your function. For example:
`ParentObj = obj.GetROProperty("parent")`

---

When writing your function, you can use standard VBScript statements as well as any QuickTest reserved objects, methods, functions, and any method associated with the test object specified in the first argument of the function.

For example, suppose you want to report the current value of an edit box to the Test Results before you set a new value for it. You can override the standard QuickTest Set method with a function that retrieves the current value of an edit box, reports that value to the Test Results, and then sets the new value of the edit box.

The function would look something like this:

```
Function MyFuncWithParam (obj, x)
    dim y
    y = obj.GetROProperty("value")
    Reporter.ReportEvent micDone, "previous value", y
    MyFuncWithParam=obj.Set (x)
End Function
```

**Note:** This function defines a return value, so that each time it is used by the component, the function returns the **Set** method argument value.

## Registering User-Defined Test Object Methods

You can use the RegisterUserFunc statement to instruct QuickTest to use your user-defined function as a method of a specified test object class for the duration of a component run, or until you unregister the method.

**To register a user-defined function as a test object method, use the following syntax:**

**RegisterUserFunc** *TOClass, MethodName, FunctionName, SetAsDefault*

| Item | Description |
|---|---|
| *TOClass* | Any test object class. |
| *MethodName* | The name of the method you want to register (and display in QuickTest, for example, in the Keyword View and IntelliSense). If you enter the name of a method already associated with the specified test object class, your user-defined function overrides the existing method. If you enter a new name, it is added to the list of methods that the object supports. |
| *FunctionName* | The name of the user-defined function that you want to call from your component. The function can be located in any function library associated with your component's application area. |
| SetAsDefault | Indicates whether the registered function is used as the default method for the test object.<br>When you select a test object in the Keyword View, the default method is automatically displayed in the **Operation** column. |

---

**Tip:** It is recommended to include the RegisterUserFunc statement in the function library so that the method will be immediately available for use in any component using that function library.

---

For example, suppose that the Find Flights Web page contains a **Country** edit box, and by default, the box contains the value USA. The following example registers the Set method to use the MySet function to retrieve the default value of the edit box before the new value is entered.

```
Function MySet (obj, x)
    dim y
    y = obj.GetROProperty("value")
    Reporter.ReportEvent micDone, "previous value", y
    MySet=obj.Set(x)
End Function

RegisterUserFunc "WebEdit", "Set", "MySet"
Browser("MercuryTours").Page("FindFlights").WebEdit("Country").Set "Canada"
```

For more information and examples, see the *HP QuickTest Professional Object Model Reference*.

## Unregistering User-Defined Test Object Methods

When you register a method using a RegisterUserFunc statement, your method becomes a recognized method of the specified test object while it is being used by the component, or until you unregister the method. If your method overrides a QuickTest method, unregistering the method resets the method to its normal behavior. Unregistering other methods removes them from the list of methods supported by the test object.

**To unregister a user-defined method, use the following syntax:**

**UnRegisterUserFunc** *TOClass, MethodName*

| Item | Description |
|---|---|
| *TOClass* | The test object class for which your method is registered. |
| *MethodName* | The method you want to unregister. |

For example, suppose that the Find Flights Web page contains a **Country** edit box, and by default, the box contains the value USA. The following example registers the Set method to use the MySet function to retrieve the default value of the edit box before the new value is entered. After using the registered method in a WebEdit.Set statement for the **Country** edit box, the UnRegisterUserFunc statement is used to return the Set method to its standard functionality.

```
Function MySet (obj, x)
    dim y
    y = obj.GetROProperty("value")
    Reporter.ReportEvent micDone, "previous value", y
    MySet=obj.Set(x)
End Function

RegisterUserFunc "WebEdit", "Set", "MySet"
Browser("MercuryTours").Page("FindFlights").WebEdit("Country").Set "Canada"
UnRegisterUserFunc "WebEdit", "Set"
```

# Additional Tips for Working with User-Defined Functions

When working with user-defined functions, consider the following tips and guidelines:

- 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.
- When you register a function, it applies to an entire test object class. You cannot register a method for a specific test object.
- If you want to call a function from additional test objects, you can copy the RegisterUserFunc line, paste it immediately after another function and replace any relevant argument values.
- It is recommended to include the RegisterUserFunc statement in the function library so that the method will be immediately available for use in any component using that function library.
- To use an Option Explicit statement in a function library associated with your component, you must include it in all the function libraries associated with the component. If you include an Option Explicit statement in only some of the associated function libraries, QuickTest ignores all the Option Explicit statements in all function libraries. You can use Option Explicit statements directly in your action scripts without any restrictions.
- Each function library must have unique variables in its global scope. If you have two associated function libraries that define the same variable in the global scope using a Dim statement or define two constants with the same name, the second definition causes a syntax error. If you need to use more than one variable with the same name in the global scope, include a Dim statement only in the last function library (since function libraries are loaded in the reverse order).
- By default, steps that use user-defined functions are not displayed in the test results tree of the Test Results window after a run session. If you want the function to appear in the test results tree, you must add a Reporter.ReportEvent statement to the function code. For example, you may want to provide additional information or to modify the component status, if required.

- If you delete a function in use from an associated function library, the component step using the function will display the icon. In subsequent run sessions for the component or business process test, an error will occur when the step using the non-existent function is reached.
- If another user modifies a function library that is referenced by a component, or if you modify the function library using an external editor (not QuickTest), the changes will take effect only after the component is reopened.
- When more than one function with the same name exists in the function library, the last function will always be called. To avoid confusion, make sure that you verify that within the resources associated with an application area or component, each function has a unique name.
- You can re-register the same method to use different user-defined functions without first unregistering the method. However, when you do unregister the method, it resets to its original QuickTest functionality (or is cleared completely if it was a new method), and not to the previous registration.

  For example, suppose you enter the following statements:

  ```
  RegisterUserFunc "Link", "Click", "MyClick"
  RegisterUserFunc "Link", "Click", "MyClick2"
  UnRegisterUserFunc "Link", "Click"
  ```

  After running the UnRegisterUserFunc statement, the Click method stops using the functionality defined in the MyClick2 function, and returns to the original QuickTest Click functionality, and not to the functionality defined in the MyClick function.
- For more information on creating functions and subroutines using VBScript, you can view the VBScript documentation from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

# Part IV

## Working with Application Areas and Components

# 12

# Working with Application Areas

Application areas provide all of the resources and settings needed to create a business component. 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.

---

**Note:** In earlier QuickTest Professional versions, the application area was known as a business component template. At that time, all business components used the same template. Now, QuickTest enables you to create multiple application areas that can be customized to suit the requirements of each area of your application.

---

**This chapter includes:**

# About Working with Application Areas

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 tested by the component, function libraries containing user-defined operations performed on that application by the component, and so forth. Components are automatically linked to all of the resources and settings defined in the associated application area.

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.

---

**Note:** To work with application areas, you must have the required permissions 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 permissions in the Business Components module, refer to the *HP Business Process Testing User's Guide*.

---

### Planning an Application Area

Before you create an application area, consider the requirements of Subject Matter Experts that will use the application area to create business components. For example:

- What test objects will they need?
- How will you rename the test objects and other items so that their meanings are clear to a wide range of users?
- What user-defined functions can you add to ensure that all required operations are available?

  To ensure availability, it is recommended that these function libraries be saved in the Quality Center project *before* creating the application area, although you can update an application area at any time. QuickTest also provides you with a set of predefined resource files that you can associate with the application area, for example, function libraries and a recovery scenario file. Some of the sample function libraries are associated with all new application areas by default. These sample files are located in the Test Plan module of your Quality Center project under **Subject/BPT Resources**.

### Creating an Application Area

When you create an application area to be used by components, you must perform the following tasks:

- Provide a full description of the application area
- Specify associations to any QuickTest Professional add-ins
- Associate any required function libraries
- Associate any required shared object repositories
- Specify which keywords will be visible and available for use by Subject Matter Experts when creating component steps
- Specify the Windows-based applications on which components associated with the application area can record and run
- Associate any required recovery scenarios and define their settings
- Save the application area

When you save the application area, make sure that you provide it with a meaningful name and a clear description. When a Subject Matter Expert creates a new business component, the name and description provide the only indication of the intended use of the application area. For example, if an application area is intended for components that test a login dialog box, you might name it "LoginDialog".

### Working with an Application Area

After you create an application area, you can notify the Subject Matter Experts so that they can begin using it to create business components. (If necessary, Subject Matter Experts can start to create a component before the application area is ready, and only later associate the application area with the component.)

If you modify resources or settings in an application area, these changes are reflected automatically in all of the business components associated with the modified application area.

If resources are used in component steps, and you later modify these resources, 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.

For this reason, it is recommended to ensure that any changes you make to an application area will not adversely affect the business components with which the application area is associated.

---

**Tip:** You can associate a component with a different application area at any time. For more information, see "Changing the Application Area Associated with a Component" on page 472.

---

# Creating an Application Area

When you create a new application area, you define all of the application area settings and resources needed to create a new business component.

---

**Note:** To create an application area, you must first connect to the Quality Center project in which you want to save the application area. This is the Quality Center project that will be used by Subject Matter Experts to define business components and create business process tests. For more information, see "Connecting to Your Quality Center Project" on page 44.

---

**To create an application area:**

Perform one of the following:

- Choose **File** > **New** > **Application Area**.
- Click the **New** button down arrow and choose **Application Area**.

---

**Tip:** If an application area is already open, clicking the **New** button opens a new application area.

---

The application area window contains several panes that enable you to specify the settings and resource files that you want business components associated with the application area to use. By associating a component with an application area, the component is automatically linked to these settings and resource files.

You can now specify the application area settings and define its resources. The table below displays information on the available options in each pane.

After you have defined the settings and resources, you can save the application area. For more information, see "Saving an Application Area" on page 451.

The application area contains the following panes, which you access by clicking the appropriate button in the sidebar:

| Pane | Contents |
|---|---|
| **General**  | Enables you to define the description and specify the associated add-ins for your application area.<br>You can also:<br>➤ Specify the Windows-based applications with which a component associated with the application area can work.<br>➤ Set the browser time-out period.<br>➤ Define recovery scenarios that specify how a component associated with the application area recovers from unexpected events and errors during a run session. For more information, see "Defining General Settings" on page 421. |
| **Function Libraries**  | Enables you to associate function libraries with your application area and to prioritize them. Also enables you to create and modify associated function libraries. For more information, see "Managing Function Libraries" on page 426. |
| **Object Repositories**  | Enables you to associate shared object repositories with your application area and to prioritize them. Also enables you to create and modify associated object repositories. For more information, see "Managing Shared Object Repositories" on page 432. |
| **Keywords**  | Enables you to determine which built-in and user-defined keywords (operations) are available to Subject Matter Experts when creating components. For more information, see "Managing Keywords" on page 439. |

# Opening an Application Area

After an application area is saved, you can open it for viewing or modification. For example, you may want to update a recovery scenario or add a function library with user-defined functions to the application area.

---

**Notes:**

To open an application area, you must first connect to the Quality Center project in which the application area is saved. For more information, see "Connecting to Your Quality Center Project" on page 44.

You cannot open an application area that was created with a later version of QuickTest on a computer running an earlier version of QuickTest.
For example, you cannot open an application area created in QuickTest 9.2 on a computer running QuickTest 8.2.

---

**To open an application area:**

**1** View the application areas connected to the current Quality Center project.

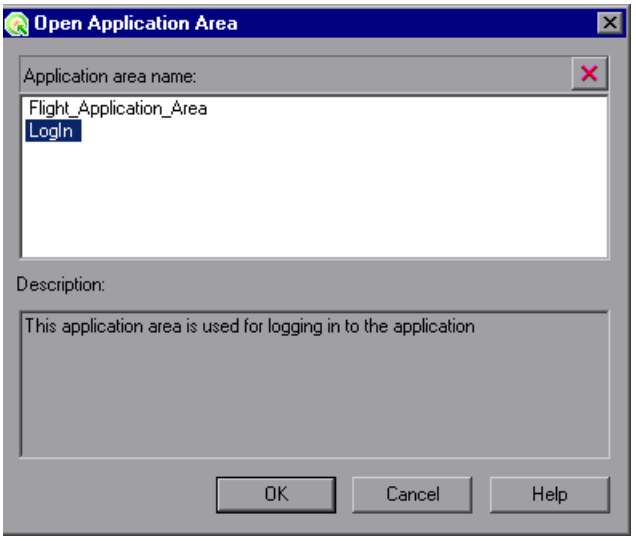
- ➤ Choose **File** > **Open** > **Application Area**.
- ➤ Click the **Open** button down arrow and choose **Application Area**.

---

**Tip:** If another application area is already open, you can click the **Open** button and then select the required application area.

---

The Open Application Area dialog box opens and displays a list of the defined application areas. You can select an application area to view its description.




**2** Select an application area and click **OK**. The selected application area opens.

You can now view and modify the settings for the application area. For more information, see:

- "Defining General Settings" on page 421
- "Managing Function Libraries" on page 426
- "Managing Shared Object Repositories" on page 432
- "Managing Keywords" on page 439
- "Defining Additional Settings" on page 443

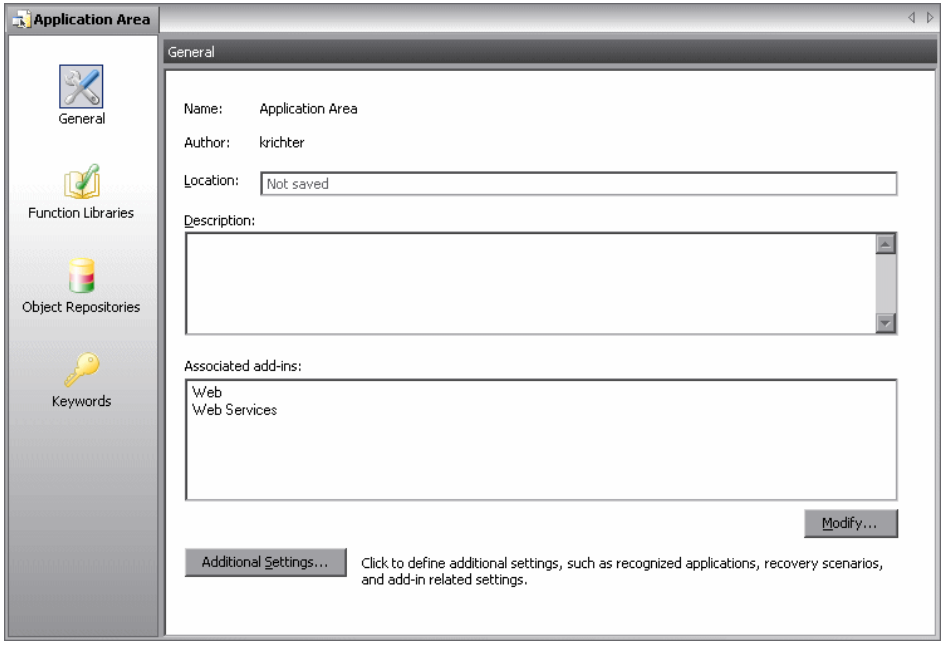
---

**Note:** You can also delete an application area from this dialog box (as long as it is not associated with any components). For more information, see "Deleting an Application Area" on page 453.

---

# Defining General Settings

You can use the General pane to view and define general information about your application area, including its description and any add-ins associated with it. It is important to include a clear description of the application area because the name and description are the only indications that a Subject Matter Expert has when determining which application area to choose for a specific business component.




The General pane includes the following items:

| Item | Description |
|---|---|
| **Name** | Indicates the name of the application area. You assign a name to the application area when you save it. For more information, see “Saving an Application Area” on page 451. |
| **Author** | Indicates the Windows user name of the person who created the application area. |

| Item | Description |
|---|---|
| **Location** | Indicates the Quality Center path and file name of the application area. If the application area is not yet saved, the location indicates **Not saved**, and the Application Area dialog box title bar contains an asterisk. |
| **Description** | Indicates the description specified for your application area.<br>It is important that this mandatory field includes a clear description of the application area. This is because the Subject Matter Expert decides which application area to choose when creating a new component in Quality Center based on the **Name** and **Description** of the application area. For more information, refer to the *HP Business Process Testing User's Guide*.<br>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**.<br>**Note:** If you do not enter a description here in the General pane, you are prompted to do so when saving the application area. For more information, see "Saving an Application Area" on page 451. |
| **Associated add-ins** | Displays the add-ins associated with the application area. The associated add-ins are those loaded by QuickTest when business components are accessed.<br>**Note:** When a business process test runs, QuickTest loads the add-ins associated with the first component in the test. Therefore, it is important to ensure that all required QuickTest add-ins are associated with the application area for the first component in the business process test.<br>For more information on associating add-ins, see "Associating Add-ins with Your Component" on page 424.<br>For more information on QuickTest add-in environments, see the *HP QuickTest Professional Add-ins Guide*. |

| Item | Description |
|---|---|
| **Additional Settings** button | Opens the Application Area Settings dialog box (described on page 443), which is divided into several tabs.<br>➤ **Applications.** Enables you to specify the Windows-based applications on which a component associated with the application area can record and run. For more information, see "Defining Application Settings for Your Application Area" on page 443.<br>➤ **Recovery.** Enables you to define how a component associated with the application area recovers from unexpected events and errors that occur in your testing environment during a run session. For more information, see "Defining Recovery Scenario Settings for Your Application Area" on page 447.<br>The Application Area Settings dialog box may also contain additional tabs corresponding to any QuickTest add-ins that are loaded, for example, Web, Java, or SAP. For information on tabs related to add-ins, see the *HP QuickTest Professional Add-ins Guide*. |
| **Modify** button | Opens the Modify Associated Add-ins dialog box. This dialog box enables you to associate add-ins with components or remove associations. You may be required to restart QuickTest for the changes to take effect. For more information, see "Associating Add-ins with Your Component" on page 424. |

---

**Note:** If you do not see the entire General pane when opening an application area, you can resize the panes. For example, if the Information pane covers the area below the associated add-ins, you can resize the Information pane.

---

## Associating Add-ins with Your Component

When you open QuickTest, 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.

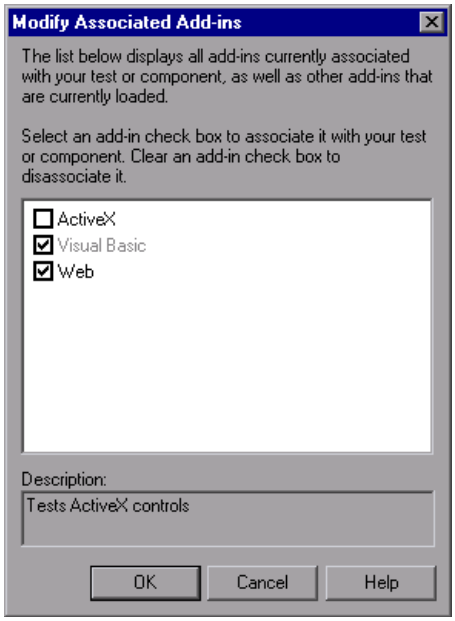
Choosing to associate an add-in with an application area instructs QuickTest to check that the associated add-in is loaded each time you open a component that is associated with that application area. When you create a new component, its associated add-ins are those defined in the component's application area.

When you open a component, QuickTest notifies you if an associated add-in is not currently loaded, or if you have loaded add-ins that are not currently associated with your 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 Subject Matter Expert opens a business process test in Quality Center, the QuickTest Professional 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 QuickTest add-ins are associated with the application area of the first component in the business process test.

## Modifying Associated Add-Ins

Click the **Modify** button in the General pane to associate or disassociate add-ins with your application area (and its associated components). The Modify Associated Add-ins dialog box opens.




This dialog box lists all the add-ins currently associated with your application area, as well as any other add-ins that are currently loaded in QuickTest. Add-ins that are associated with your application area but not currently loaded are shown dimmed.

---

**Note:** 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. For more information, see the relevant Add-in Extensibility Developer's Guide (available with the extensibility setup).

---

You can select the check boxes for add-ins that you want to associate with your application area, or clear the check boxes for add-ins that you do not

want to associate with your application area. If the Modify Associated Add-ins 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.

In the above example:

- Web is loaded and associated with the application area.
- ActiveX is loaded, but not associated with the application area.
- Visual Basic is associated with the application area, but is not loaded.

---

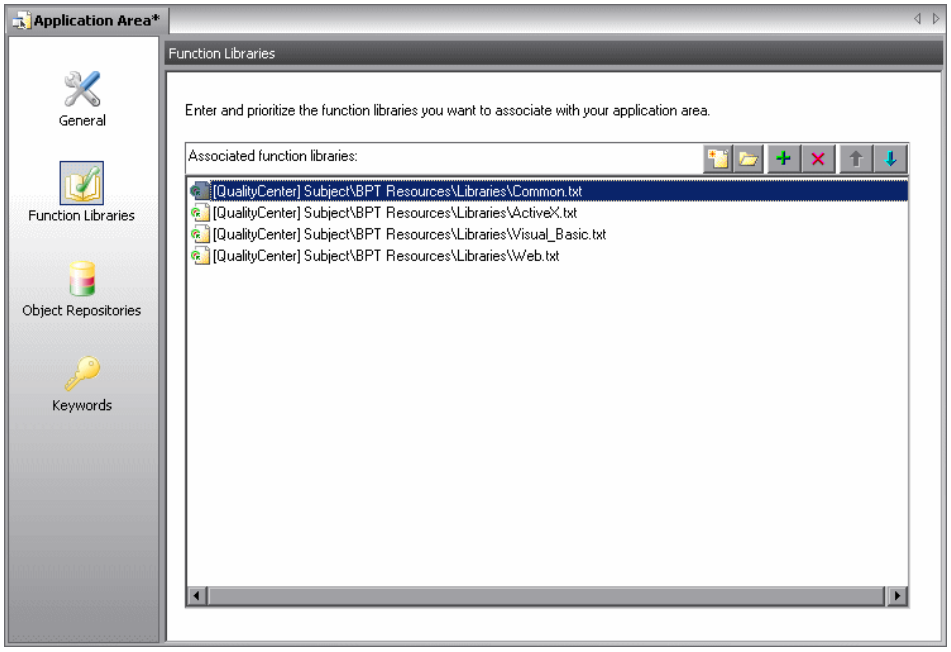
**Note:** If a specific add-in is not currently loaded, but you want to associate it with an application area, reopen QuickTest and load the add-in from the Add-in Manager. If the Add-in Manager dialog box is not displayed when you open QuickTest, you can choose to display it the next time you open QuickTest. To do so, select **Display Add-in Manager on startup** from the General tab of the Options dialog box. For more information, see "Setting General Testing Options" on page 584.

---

## Managing Function Libraries

In the Function Libraries pane, you can associate function library files, such as QuickTest function libraries, VBScript function libraries, or text files, with your application area. You associate function libraries with your application areas to provide additional functionality in the form of user-defined keywords that can be used when creating business components.

All associated function libraries must be saved in your Quality Center project.




The Function Libraries pane displays the list of function libraries currently associated with your application area and enables you to associate additional function libraries, and to 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 Quality Center project.

---

**Note:** QuickTest 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 your Quality Center project, under **Subject\BPT Resources\Libraries**. For information on creating user-defined functions in function libraries, see Chapter 11, "Working with User-Defined Functions and Function Libraries."

---

You can add, modify, delete, and prioritize function libraries associated with your component using the following buttons:

| Button | Description |
|---|---|
|  | Enables you to create a new function library, save it to your Quality Center project, and add it to the list. |
|  | Opens the selected function library for viewing or editing in a function library window. Function libraries that are currently in use by another QuickTest or Quality Center user are locked and can be opened only in read-only mode. For more information, see Chapter 11, "Working with User-Defined Functions and Function Libraries." |
|  | Enables you to browse to the test plan tree of your Quality Center project and select an existing function library to associate with the application area. For more information, see "Associating Existing Libraries with Your Application Area" on page 429. |
|  | 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. |
|  | Moves the selected function library down in the list, giving it a lower priority during the component run session. |

---

**Note:** You can right-click an associated function library and choose **Open** to open it, or **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 Quality Center project, QuickTest 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 choose **Locate** to browse to the required function library, or **Remove** to remove the association to the missing function library.

---

## Associating Existing Libraries with Your Application Area

You can add existing function libraries to your application area. This enables all business components associated with application area to access the functions defined in these function libraries as keywords.

**To associate an existing function library with the application area:**

**1** In QuickTest, open the application area (if it is not already open).

- ➤ Choose **File** > **Open** > **Application Area**.
- ➤ Click the **Open** button down arrow and choose **Application Area**.

---

**Tip:** If another application area is already open, you can click the **Open** button and then select the required application area.

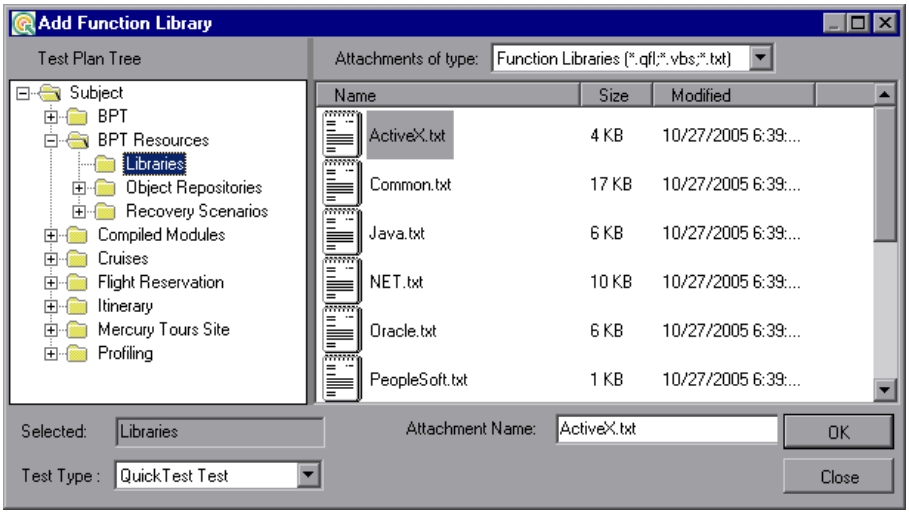
---

**2** Click **Function Libraries** in the sidebar. The list of function libraries currently associated with the application area is displayed in the Function Libraries pane.

**3** Click the **Add Function Library** button. A blank line is added to the list, as well as a browse button.

**4** Click the browse button. The Add Function Library dialog box opens.

The Add Function Library dialog box displays the test plan tree of the current Quality Center project.




5 Select the relevant item in the tree to display its attached function libraries. Then select the function library that you want to associate with your application area. The name is displayed in the **Attachment Name** box.

6 Click **OK**. The Add Function Library dialog box closes and the selected file is displayed in the Function Libraries pane of the application area. If the function library contains syntax errors, a message opens stating that your test will fail because of these syntax errors.

## Creating New Function Libraries

You can create new function libraries directly from the Function Libraries pane of the application area and associate them automatically to your application area.

**To create a new function library in your Quality Center project:**

1 In QuickTest, open the application area (if it is not open).

➤ Choose **File** > **Open** > **Application Area**.

➤ Click the **Open** button down arrow and choose **Application Area**.

---

**Tip:** If another application area is already open, you can also click the **Open** button and then select the application area you require.

---

**2** Click **Function Libraries** in the sidebar. The list of function libraries currently associated with the application area is displayed in the Function Libraries pane.

**3** In the Function Libraries pane, click the **Create Function Library** button. The Add Function Library dialog box opens.

**4** In the test plan tree, navigate to the folder in which you want to store the function library.

**5** In the **Attachment Name** box, enter a name for the function library and click **OK**. A new empty function library is added to the selected location in the test plan tree and listed in the Function Libraries pane.

---

**Note:** By default, function libraries are created in QuickTest as **.qfl** files if no suffix is specified. You can also create **.txt** or **.vbs** files if needed.

---

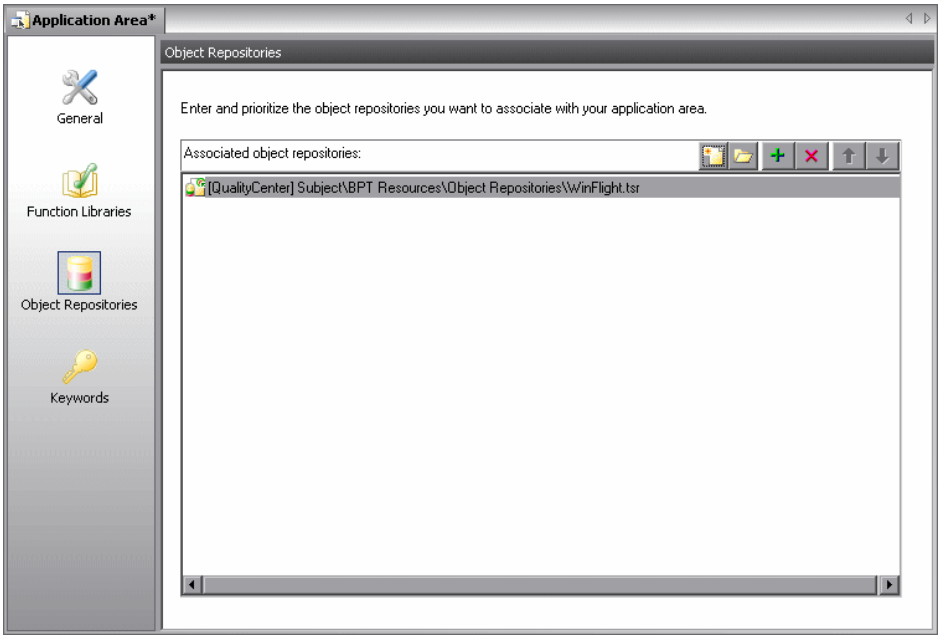
**6** If you want to add content to the new function library or modify the file directly from QuickTest, select the file in the Function Libraries pane and click the **Open Function Library** button or double-click the function library in the list. The file opens in a function library window and can be edited as required. To save your changes, close the file and click **Yes** when prompted.

**7** If you want to rename the function library, you can click it twice, or select it and press F2.

For more information on editing function libraries, see Chapter 11, “Working with User-Defined Functions and Function Libraries.”

# Managing Shared Object Repositories

A shared object repository stores all of the test objects that may be used when creating steps for a business component. After you associate a shared object repository with an application area, it can be accessed by any component that is associated with that application area.




The Object Repositories 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 add existing object repositories or create new ones, as long as the object repositories are stored in your Quality Center project.

You can add test objects to this shared object repository either by learning objects in your application or by adding test objects manually using the Object Repository Manager. For information on managing test objects in a shared object repository, see Chapter 6, “Managing Object Repositories.”

---

**Note:** Although QuickTest provides you with a default shared object repository (located in the **Subject/BPT Resources/Object Repositories** folder), it is strongly recommended not to use it. If you associate this default shared object repository with an application area or a specific component, any components using this shared object repository may not run correctly.

---

You can use existing shared object repositories that already contain your test objects, or you can create new ones. All business components associated with an application area that refers to these shared object repositories will then access these shared object repository files. For more information, see "Creating New Shared Object Repositories" on page 435.

After you add test objects to the shared object repository, you and Subject Matter Experts can then use the test objects to add steps to business components. For more information, see "Selecting an Item for Your Step" on page 516.

Subject Matter Experts need to be able to distinguish between the various test objects when they define steps for a business component. Therefore, it is important that all test object names be self-explanatory. You can change the name that QuickTest assigns automatically to a stored test object. For example, if a test object is named Edit by default, you may want to rename it to UserName (if that is what the user needs to enter in the edit box, of course).

For container objects, it is recommended to specify their context, for example, if you have several confirmation message boxes, you may want to name one Login > Confirm, another ChangePassword > Confirm, and still another BillingInfo > Confirm.

When you modify the name of an object, the name is automatically updated in the QuickTest Keyword View and the Steps tab of the Quality Center Business Components module for all occurrences of the object (also in steps that were created using the old test object names). When you open another component that uses the same shared object repository and has one or more occurrences of the modified object, the names within that component are updated. This may take a few moments.

For more information on renaming test objects, see “Renaming Test Objects” on page 140.

You can add, modify, delete, and prioritize object repositories associated with an application area (and its associated components) using the following buttons:

| Button | Description |
|---|---|
|  | Enables you to create a new object repository, save it to Quality Center, and then add it to the list. |
|  | Opens the selected object repository for viewing or editing in the Object Repository Manager. Object repositories that are currently locked are opened in read-only format. For more information on the Object Repository Manager, see Chapter 6, “Managing Object Repositories.” |
|  | Enables you to browse to the test plan tree of your Quality Center project and select an existing object repository to associate with the application area. For more information, see “Adding Existing Shared Object Repositories to Your Application Area” on page 437. |
|  | 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. |

---

**Note:** You can right-click a shared object repository and choose **Open** to open it in the Object Repository Manager, or **Remove** to remove its association with the application area.

If a shared object repository cannot be found, QuickTest displays its name and path in the Missing Resources 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 choose **Locate** to browse to the required shared object repository, or **Remove** to remove the association to the shared object repository.

---

## Creating New Shared Object Repositories

To enable a Subject Matter Expert to access the test objects from the application when implementing component steps, the test objects must be stored in a shared object repository located in your Quality Center project. You can create new shared object repositories directly from the Object Repositories pane of the Application Area and associate them automatically to your application area.

**To create a new shared object repository in your Quality Center project:**

**1** In QuickTest, open the application area (if it is not open).

- ➤ Choose **File** > **Open** > **Application Area**.
- ➤ Click the **Open** button down arrow and choose **Application Area**.

---

**Tip:** If another application area is already open, you can also click the **Open** button and then select the application area you require.

---

2 Click **Object Repositories** in the sidebar. The list of object repositories currently associated with the application area is displayed in the Object Repositories pane.

3 In the **Object Repositories** pane, click the **Create Object Repository** button. The Add Object Repository dialog box opens, showing the test plan tree of the current project.

4 In the test plan tree, navigate to the folder in which you want to store the object repository.

5 In the **Attachment Name** box, enter a name for the object repository and click **OK**. A new object repository is added to the selected location in the test plan tree and listed in the Object Repositories pane.

6 If you want to add test objects to your shared object repository or modify the file directly from QuickTest, select the file in the Object Repositories pane and click **Open Object Repository** or double-click the object repository in the list. The file opens in the Object Repository Manager and can be edited as required.

7 If you want to rename the object repository, you can click it twice, or select it and press F2.

For more information on modifying object repositories in the Object Repository Manager, see Chapter 6, “Managing Object Repositories.”

## Adding Existing Shared Object Repositories to Your Application Area

You can add existing shared object repository files to your application area. This enables all business components with which the application area is associated to access the test objects that are stored in these files.

**To add an existing shared object repository for the application area:**

**1** In QuickTest, open the application area, (if it is not open).

➤ Choose **File** > **Open** > **Application Area**.

➤ Click the **Open** button down arrow and choose **Application Area**.

---

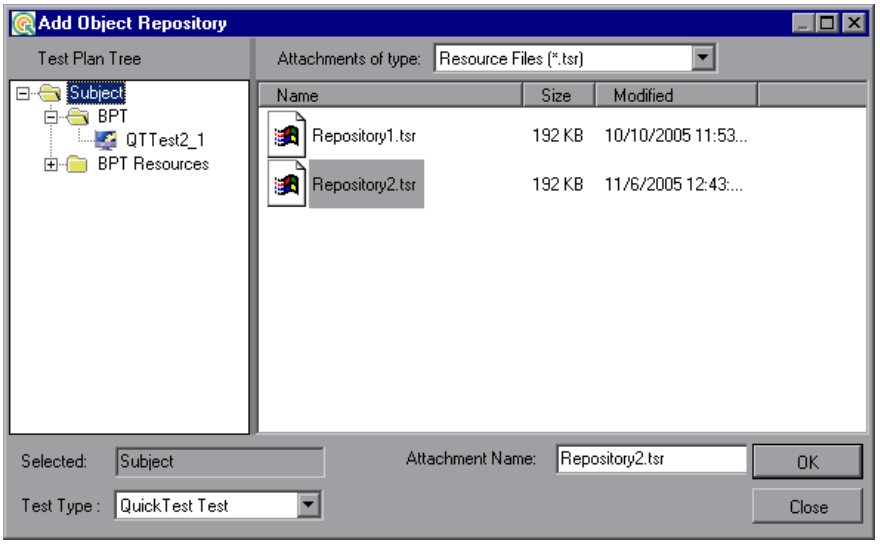
**Tip:** If another application area is already open, you can also click the **Open** button and then select the application area you require.

---

**2** Click **Object Repositories** in the sidebar. The list of object repositories currently associated with the application area is displayed in the Object Repositories pane.

**3** Click the **Add Object Repository** button. A blank line is added to the list, as well as a browse button.

**4** Click the browse button. The Add Object Repository dialog box opens. The dialog box displays the test plan tree of the current Quality Center project.




**5** Select the relevant item in the tree to display its attached object repositories. Then select the object repository that you want to associate with your application area. The name is displayed in the **Attachment Name** box.

**6** Click **OK**. The Add Object Repository dialog box closes and the selected file is displayed in the Object Repositories pane of the application area.

**7** If you want to add test objects to your shared object repository or modify the file directly from QuickTest, select the file in the Object Repositories pane and click **Open Object Repository** or double-click the object repository in the list. The file opens in the Object Repository Manager and can be edited as required.

**8** If you want to rename the object repository, you can click it twice, or select it and press F2.

# Managing Keywords

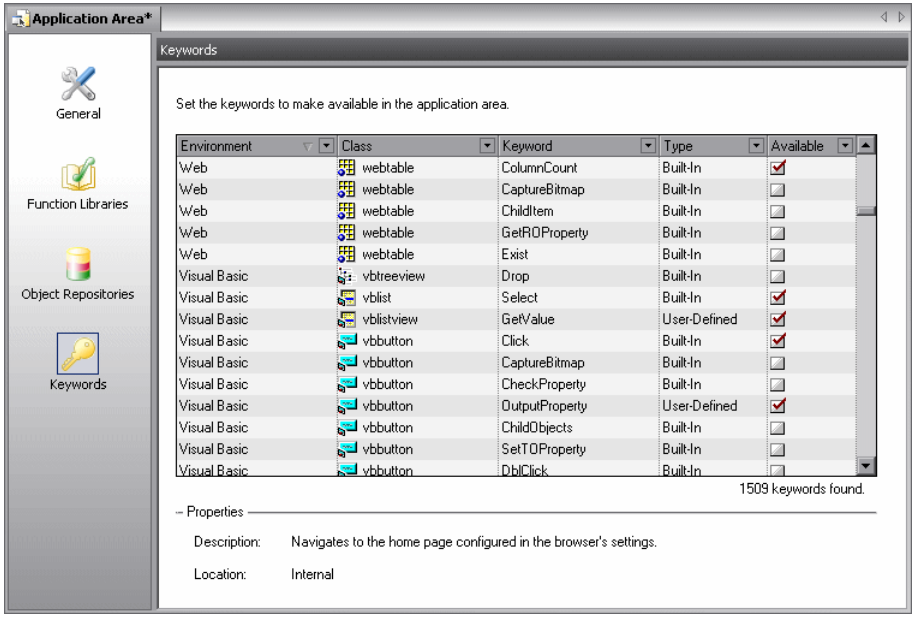
When creating a step in a business component, Subject Matter Experts select the required operation to perform on the application being tested. These operations are also known as keywords, and are derived from built-in methods and properties, as well as user-defined functions associated with the application area.

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. The Keywords pane enables you to manage the keywords and select which of them should be available to Subject Matter Experts when creating business components. Only selected built-in keywords are available by default. However, all user-defined keywords are available to Subject Matter Experts.

---

**Note:** The Keywords pane is not relevant for scripted components.

---



To make keywords available to Subject Matter Experts from the lists of operations in business component steps, click the relevant check boxes in the **Available** column. To remove keywords from the lists of available operations, clear the check boxes. Subject Matter Experts will not be able to use keywords whose check boxes are cleared.

The Keywords pane displays information about the keywords in the following columns:

| 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.<br>**Notes:**<br>➤ 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.<br>➤ 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** | Whether the operation is Built-in (provided by QuickTest) or User-Defined (contained in a function library). |
| **Available** | Whether the keyword is available to Subject Matter Experts for use in business component steps. You can select or clear each check box as required. |

Clicking a keyword in the list displays information about it in the **Properties** area at the bottom of the pane. This includes a textual description of what the keyword does, as well as the name and path of its function library (for user-defined keywords). The location of a built-in keyword is defined as Internal.

You can view, sort, and filter the data in the Keywords pane to make it easier to locate the keywords that you want to make available to (or hide from) the Subject Matter Experts.

---

**Tips:**

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.

---

## Filtering the Columns

You can filter the data in the Keywords pane 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.

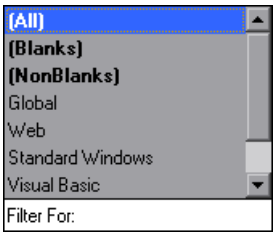
For example, you may want to view only Web Add-in keywords that are currently not available to Subject Matter Experts. You would filter the **Environment** column to display only keywords from the Web Add-in, and then filter the **Available** column to display only keywords whose check box is cleared (select **Unchecked** from the **Available** column filter list).

The filter criteria and the number of keywords that match the current filter are displayed below the columns.


Click the ✕ to the left of the filter criteria to clear the filter and show all keywords.

**To filter the data in a column:**

Click the arrow in a column header. A list of the unique items contained in the column opens.




You can perform the following to filter the data in the column:

- 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 choose an item in that list. The filtered data is filtered again to show only the keywords that match all selected filter criteria.

- 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 a 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.

### Sorting Column Content

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.

## Defining Additional Settings

Clicking the **Additional Settings** button in the General pane opens the Application Area Settings dialog box, which comprises several tabs. These tabs enable you to define specific settings for your application area, such as the applications on which the components associated with the application area can record and run, and how a component recovers from unexpected events during a run session.

### Defining Application Settings for Your Application Area

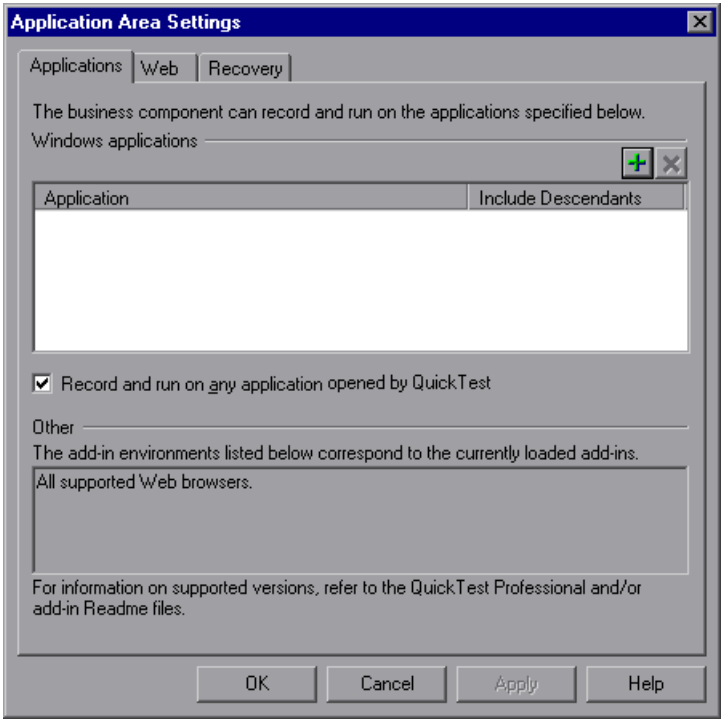
In the Applications tab, you can specify the Windows-based applications on which the components associated with this application area can record and run. You can record component steps only on the specified applications.

---

**Tip:** To record on an application, you can either open it manually, or you can use the OpenApp keyword (function) provided with QuickTest in the **Common.txt** function library. There are no settings available for automatically opening applications for components.

---

The **Other** area displays the environments on which the application area's associated components can currently record (based on the currently loaded add-ins).




You can use the Applications tab 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.

**Notes:**

If you are recording a new component and have not yet set your application settings in the Applications tab of the Application Area Settings dialog box, the Applications dialog box opens when you start to record. The Applications dialog box contains the same options as the Applications tab, described in this section.

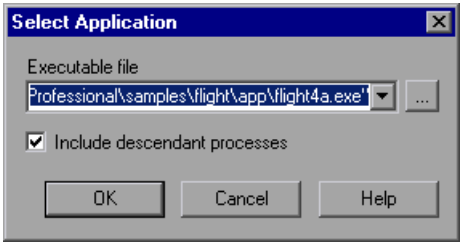
The Applications dialog box and Applications tab may also contain options applicable to any QuickTest add-ins installed on your computer. For information regarding these options, refer to the documentation provided for the specific add-in.

The following options are available in the Applications tab:

| Option | Description |
|---|---|
| **Windows applications** | Lists the details of the applications on which to record and run components associated with this application area. For more information on the details displayed, see "Specifying an Application" on page 446.<br>If you do not want to record or run on Windows 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 more information, see "Specifying an Application" on page 446. |
|  | Removes the selected application from the application list. |
| **Record and run on any applications opened by QuickTest** | Records and runs on any applications invoked by QuickTest (as child processes of QuickTest). For example, applications opened during a record or run session using an OpenApp function, or another operation containing a function that opens an application. |
| **Other** | Lists the add-in environments that correspond to the currently loaded add-ins. |

## Specifying an Application

When you click the **Add** button in the Applications tab, the Select Application dialog box opens.




You can add up to ten applications to the application list displayed in the Applications tab, 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 applications** area of the Applications tab.

You can specify the following details for the application in the Select Application dialog box:

| Option | Description |
|---|---|
| **Executable file** | Instructs QuickTest to record and run on the specified executable file. |
| **Include descendant processes** | Selecting this check box instructs QuickTest to record and run on processes created by the specified application during the record and run session. For example, a process that is used only as a launcher may create another process that actually provides the application functionality. This descendant process must therefore be included when recording or running tests on this application, otherwise the functionality will not be recorded, or the run session will fail.<br>By default, this option is selected. |

## Defining Recovery Scenario Settings for Your Application Area

Recovery scenario settings enable you to specify how a business component recovers from unexpected events and errors during a run session.

The Recovery tab displays a list of all recovery scenarios associated with the current application area. It also enables you to associate additional recovery scenarios with the application area, remove scenarios from the application area, change the order in which they are applied to the run session, and view a read-only summary of each scenario.

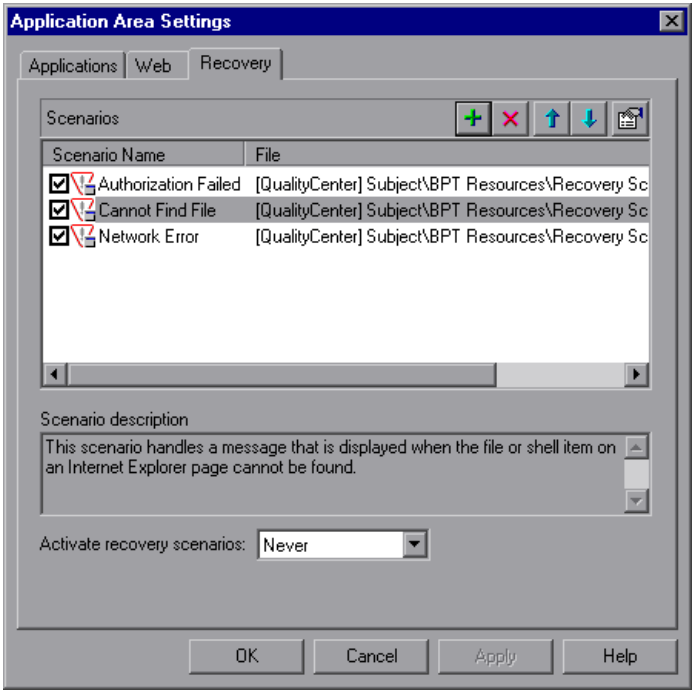
You can enable or disable specific scenarios or the entire recovery mechanism for the application area. You can add existing recovery scenarios or create new ones, as long as the recovery scenarios are stored in your Quality Center project.

---

**Note:** QuickTest provides you with a sample recovery file for Web-related testing. The file is located in your Quality Center project, under **Subject\BPT Resources\Recovery Scenarios\DefaultWeb.qrs**.

---

You define recovery scenarios for application areas in exactly the same way as for tests. For more information on recovery scenarios, see Chapter 32, “Defining and Using Recovery Scenarios.”



The Recovery tab includes the following options:

| Option | Description |
|---|---|
| **Scenarios** | Displays the name and recovery file path for each recovery scenario associated with your application area. You can add, delete, and prioritize the scenarios in the list, and you can edit the file path for a selected file. For more information, see "Specifying Associated Recovery Scenarios" on page 449. |
| **Scenario description** | Displays the textual description of the scenario selected in the **Scenarios** box. |
| **Activate recovery scenarios** | Instructs QuickTest to check when to run the associated scenarios as follows:<br>➤ **On every step.** The recovery mechanism is activated after every step. (Note that choosing **On every step** may result in slower performance during the run session.)<br>➤ **On error.** The recovery mechanism is activated only after steps that return an error return value.<br>➤ **Never.** The recovery mechanism is disabled. |

## Specifying Associated Recovery Scenarios

You can select or clear the check box next to each scenario to enable or disable it for the current application area.

You can also edit the recovery scenario file path by clicking the path once to highlight it, and then clicking it again to enter edit mode. If you modify a recovery scenario file path, ensure that the recovery scenario exists in the new path location before running components that are associated with this application area.

Scenario types are indicated by the following 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 component does not run successfully. |
|  | Indicates that the recovery scenario is triggered when a specified application fails during the run session. |
|  | Indicates that the recovery scenario is no longer available for the application area. This may be because the recovery file has been renamed or moved, or can no longer be accessed by QuickTest. When an associated recovery file is not available during a run session, a message is displayed in the results. |

You can add, delete, and prioritize the recovery scenario files associated with your component using the following buttons:

| Button | Description |
|---|---|
|  | Opens the Add Recovery Scenario dialog box, which enables you to associate one or more recovery scenarios with the component. For more information, see Chapter 32, “Defining and Using Recovery Scenarios.” |
|  | Removes the selected recovery scenario from the component. |
|  | Moves the selected scenario up in the list, giving it a higher priority during the component run session. |
|  | Moves the selected scenario down in the list, giving it a lower priority during the component run session. |
|  | Displays summary properties for the selected recovery scenario in read-only format. For more information, see Chapter 32, “Defining and Using Recovery Scenarios.” |

# Saving an Application Area

You can save an application area before or after you define its settings and resources.

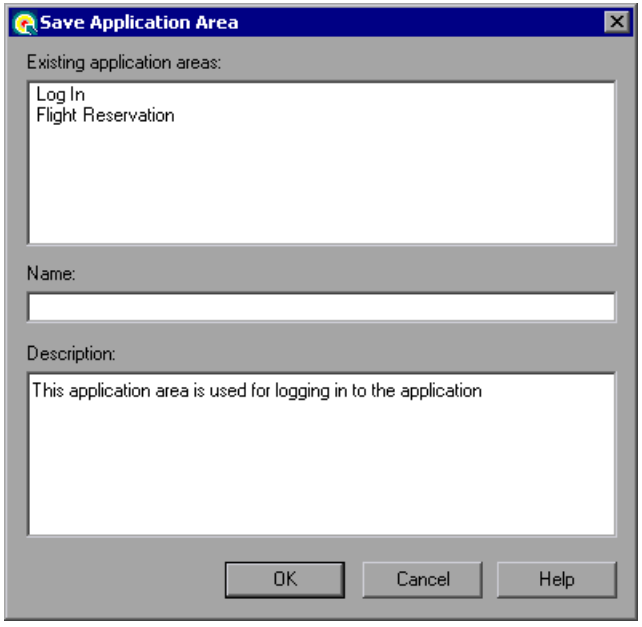
When you save an application area, make sure that you provide a unique name and description that clearly indicate its use. For example, if the application area is intended to be used by components that test the Login module, you might name it "Log In" and add a description that specifies its intended use, such as, "Intended for use with business components that test the Login module."

**To save an application area:**

1 In QuickTest, connect to a Quality Center server and project with Business Process Testing support. For more information, see "Connecting to Your Quality Center Project" on page 44.

2 Create an application area and modify its settings as required. For more information, see "Creating an Application Area" on page 417.

3 Click **Save** or choose **File** > **Save**. The Save Application Area dialog box opens.



The Save Application Area dialog box includes the following:

| Option | Description |
|---|---|
| **Existing application areas** | Lists all defined application areas in the Quality Center project. This enables you to see the names of the existing application areas so that you can specify a unique name for the application area that you want to save. |
| **Name** | Indicates the name of the application area. Enter a descriptive name that will enable Subject Matter Experts to quickly identify the application area that is suitable for their component.<br>**Note:** The name you enter cannot exceed 220 characters, cannot contain begin or end with spaces, and cannot contain the following characters:<br>\ / : " ? < > \| * ! { } ‘ % ; |
| **Description** | Displays the description you entered in the General pane of the Application Area dialog box when you created the application area. For more information, see “Creating an Application Area” on page 417.<br>If you did not enter a description when you created the application area, you must enter one now. You cannot save an application area without a description.<br>You can also modify the existing description if you already defined one in the General pane. The description you provide enables Subject Matter Experts to easily differentiate between the various application areas and choose the one that is best suited for their component. |

**4** Enter the required information and click **OK** to save the application area.

---

**Tip:** If you are creating a new application area that is similar to an existing one, you can use the **Save As** option. Then you can modify the application area, as needed.

---

# Deleting an Application Area

If an application area is no longer needed, you can delete it. Before you delete an application area, you must make sure that it is not being used by any business components. You cannot delete an application area that is used by a business component.

**To delete an application area:**

1 In QuickTest, connect to the Quality Center project that contains the application area that you want to delete. For more information, see "Connecting to Your Quality Center Project" on page 44.

2 In QuickTest, open the application area, (if it is not open).

➤ Choose **File** > **Open** > **Application Area**.

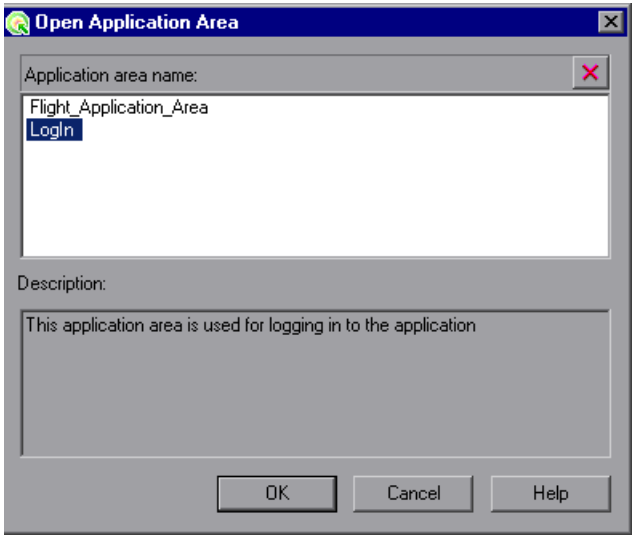
➤ Click the **Open** button down arrow and choose **Application Area**.

---

**Tip:** If another application area is already open, you can also click the **Open** button and then select the application area you require.

---

The Open Application Area dialog box opens.



**3** Select the application area that you want to delete and click the **Delete Application Area** button. A warning message displays.

---

**Note:** You cannot delete the currently open application area, an application area that is currently being used by another Automation Engineer, or an application area that is associated with a component.

---

**4** Click **Yes** to confirm. The selected application area is deleted.

**5** Click **OK** to close the Open Application Area dialog box.

# 13

# Working with Business Components

You can use the Business Component Keyword View to create, view, modify, and debug a business component in QuickTest.

**This chapter includes:**

# About Working with Business Components

Generally, business components are created and modified in Quality Center by Subject Matter Experts. For more information, see the *HP Business Process Testing User's Guide*. However, you can use the Business Component Keyword View to create, view, modify, and debug a business component in QuickTest, if required.

In the Keyword View, business 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 by selecting items and operations and entering additional information, as required.

Each step in a business 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 business component, when you define a step by selecting a user-defined operation (function), the documentation that you added in the function library will be displayed for the step. For more information, see "Documenting the Function" on page 399.

Before you create or open a business component, you connect QuickTest to a Quality Center project, which is where business components and application area resources and settings are stored. Connecting to your Quality Center project enables QuickTest to create or open the business component. This also enables the business component to access all of the resources defined in the application area on which the component is based.

---

**Note:** You need to make sure you have the required Quality Center permissions before working with business components and application areas. For more information on setting user group permissions in the Business Components module, refer to the *HP Business Process Testing User's Guide*.

---

If the application area you select does not yet contain all of the required resources and settings, you can still add steps using the ManualStep function or the **Comment** option. This enables you to type in manual steps as you would in Quality Center or in another application, such as Microsoft Excel or Microsoft Word. You can also use comments to add information about a step or to separate sections of your business component. Each manual step and comment appears as a separate row in the Keyword View. For more information, see "Adding and Modifying Manual Steps for Components" on page 471 and "Working with Comments" on page 541.

---

**Notes:**

If you want to delete a component, you can do so only in Quality Center, regardless of whether it was created in QuickTest or in Quality Center. For more information, refer to the *HP Business Process Testing User's Guide*.

If needed, you can convert a business component to a scripted component. For more information, see Chapter 14, "Creating Scripted Components."

---

# Creating a New Business Component

When QuickTest is connected to a Quality Center project, you can create a new business component in that project.

Each business component is based on a specific application area, which is stored in the Quality Center project in which you intend to save the component. Each application area specifies the settings and resources for the business 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 business component. For more information, see Chapter 12, "Working with Application Areas."

Generally, business components are created in Quality Center by Subject Matter Experts. For more information, refer to the *HP Business Process Testing User's Guide*. However, you can also create business components in QuickTest, if needed. This section describes how to create a new component in QuickTest.

---

**Note:** To create a new business component in QuickTest, you must have the necessary permissions to create a business process test. For more information, refer to the *HP Quality Center Administrator's Guide*.

---

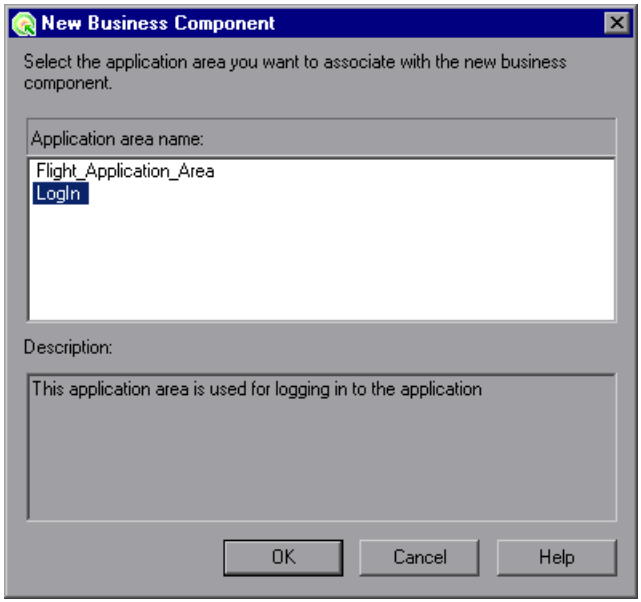
**To create a new business component:**

1 Connect to the Quality Center project in which you want to save the business component. For more information, see "Connecting to Your Quality Center Project" on page 44.

2 Perform one of the following:

- Choose **File** > **New** > **Business Component**.
- Click the **New** button down arrow and choose **Business Component**.

The New Business Component dialog box opens, listing all available application areas. You can click on an application area to view its description. (These are the descriptions that Subject Matter Experts use to determine which application area to choose when they create a new business component.)



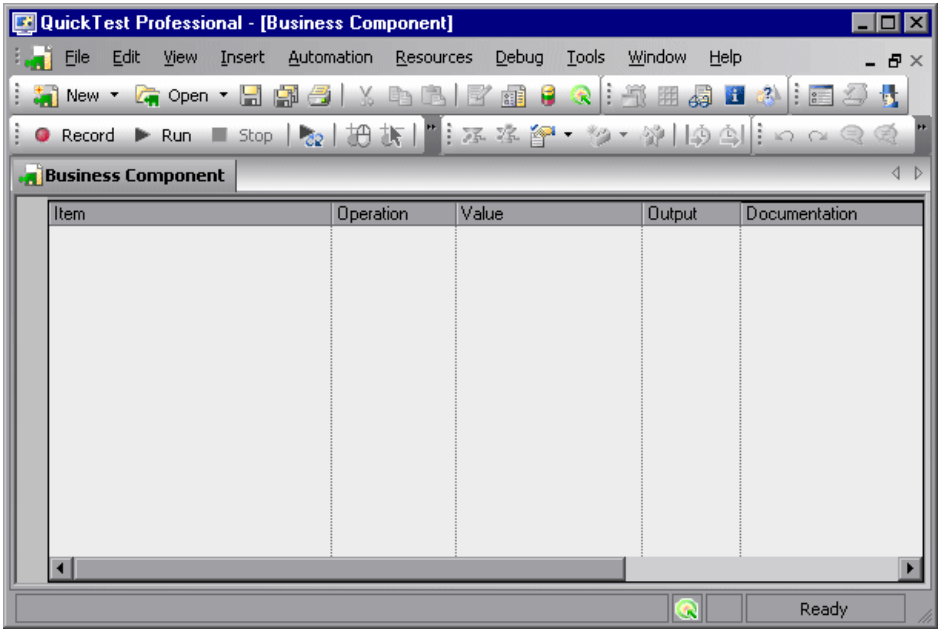

---

**Note:** If you have not yet defined an application area, a new, untitled business component opens using the default settings that are supplied with Business Process Testing. Later, after you define an application area, you can base the business component on it. For more information, see Chapter 12, "Working with Application Areas."

---

**3** Select a suitable application area from the **Application Area name** box. For example, if you want to create a business component for a Log In module, select the application area that is defined for it. Click **OK**.

A new, untitled business component opens in the Keyword View. Although the business 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** You can now:

- Add steps and comments to your business component. For more information, see "Adding a Step to Your Component" on page 514 and "Working with Comments" on page 541.
- Save your component. (You can add steps later.) For more information, see "Saving a Business Component" on page 464.

# Opening a Business Component

When QuickTest is connected to a Quality Center project, you can open a component that is stored in the project to view, modify, debug, or run it. You find components according to their location in the component tree.

Components that are currently open in Quality Center or another QuickTest session are locked and can be opened only in read-only format. To work with these components, they must be closed everywhere else.

When you open a component, if the component's associated application area cannot be found, you are prompted to associate a different application area with it.

---

**Notes for users of previous QuickTest versions:**

- When you open a business component or scripted component that was created using an earlier version of QuickTest, you are asked whether you want to convert it or view it in read-only format. If you choose to view it in read-only format, it appears as it did previously, using all of its original settings, but you cannot modify it. If you choose to convert it, it is updated to the current format. When the component is updated, it uses the associated application area's current settings. If the component had customized settings (settings that were defined directly in the Business Component Settings dialog box), these settings are removed and the associated application area's current settings are applied instead.
- After you save a converted component, it cannot be used with earlier versions of QuickTest.
- You cannot open a component that was created with a later version of QuickTest on a computer running an earlier version of QuickTest. For example, you cannot open a component created in QuickTest 9.2 on a computer running QuickTest 8.2.

---

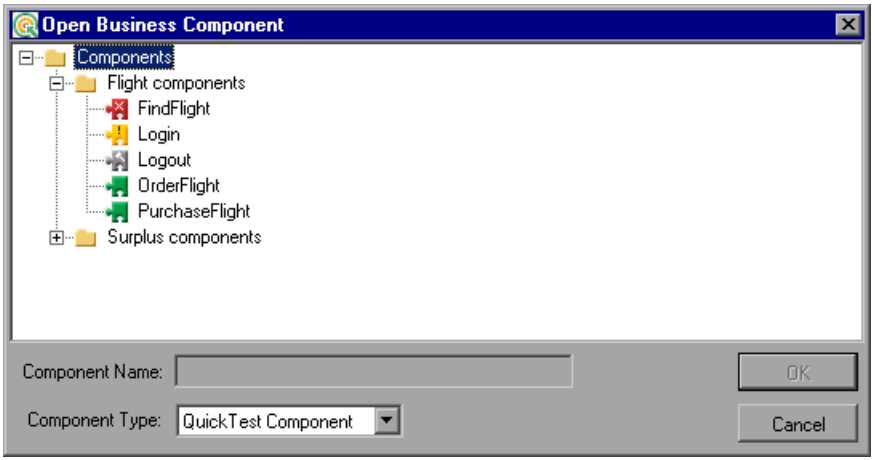
You open business components and scripted components in the same way. For more information on scripted components, see "Creating Scripted Components" on page 475.

**To open an existing component:**

1 In QuickTest, connect to the Quality Center project in which your component is saved. For information on connecting to Quality Center, see "Connecting to Your Quality Center Project" on page 44.

2 Choose **File** > **Open** > **Business/Scripted Component**, or click the **Open** down arrow and choose **Business/Scripted Component**. The Open Business Component dialog box opens showing the components stored in the Quality Center project.

You can change the type of components displayed in the dialog box, as described in step 3.




In the component tree, the status of each component is indicated by its icon. For more information, refer to the *HP Business Process Testing User's Guide*.

---

**Tip:** You can also open a recently used component by selecting it from the Recent Files list in the **File** menu. If you select a component when you are not connected to the Quality Center project, or if you select a component that is stored in a different Quality Center project, QuickTest displays a message asking you if you want to connect to that project. For more information, see "Opening Components from the Recent Files List" on page 464.

---

**3** If required, filter the list of components shown in the Business Component dialog box by selecting the component type you want to open from the **Component Type** list. You can select one of the following component types:

- **QuickTest Component:** Displays components that were automated using QuickTest Professional or the Business Process Testing Keyword View.
- **Manual Component:** Displays components that were created in Quality Center and have not yet been converted to automated components. For more information, see "Working with Manual Components" on page 467.
- **All Components:** Displays all QuickTest automated components and manual components.

**4** Click the relevant folder in the component tree. To expand the tree and view the business components, double-click closed folders. To collapse the tree, double-click open folders.

**5** Select a component. The component name is displayed in the read-only **Component Name** box.

**6** Click **OK** to open the component.

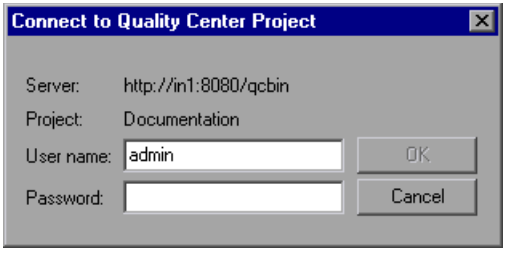
As QuickTest downloads and opens the component, the operations it performs are displayed in the status bar.

When the component opens, the QuickTest title bar displays Components, the full path and the component name. For example, the title bar for a flight_login component may be:

[Components\Flight\flight_login]

### Opening Components from the Recent Files List

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




Log in to the project, and click **OK**.

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

## Saving a Business Component

After you create or modify a component, you can save it to your Quality Center project. When you save a component, you give it a descriptive name and save it to the relevant folder in the component tree in the Quality Center project (Business Components module).

You can also save a copy of an existing component to any folder in the same Quality Center project. To enable all users to differentiate between the various components, you may want to rename a copy of a component, even if you save it to a different folder.

---

**Tip:** If changes are made to a component, an asterisk is displayed in the title bar until the component is saved.

---

You save business components and scripted components in the same way. For more information on scripted components, see "Creating Scripted Components" on page 475.

You can also convert a business component to a scripted component. For more information, see "Converting a Business Component to a Scripted Component" on page 482.

---

**Note:** For scripted components only, the data sheet name in the Data Table is identical to the scripted component name. If you save a scripted component with a new name (**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 Expert View and replace them with the new data sheet name.

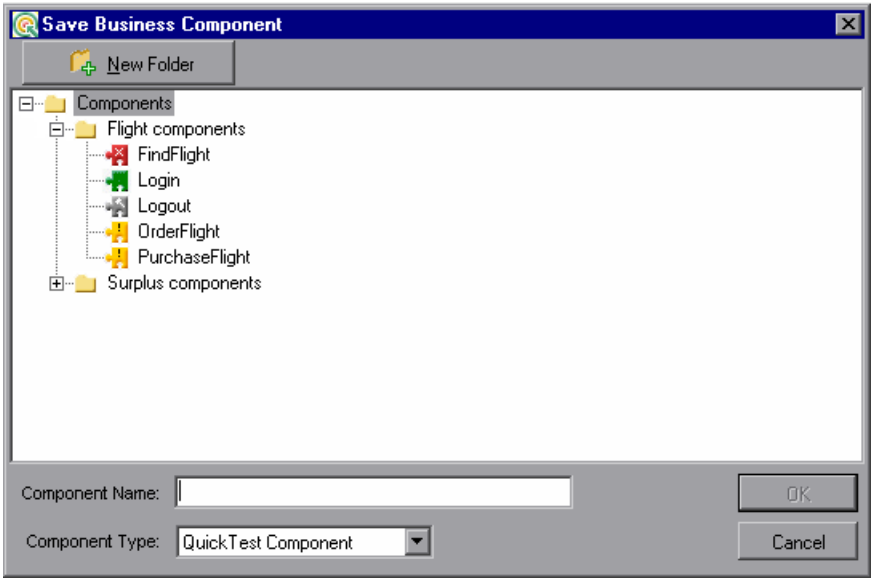
---

**To save a component to your Quality Center project:**

**1** Save the component in one of the following ways:

- For a new component that has never been saved, choose **File** > **Save** or click **Save**.
- To save a copy of an existing component, choose **File** > **Save As**.

The Save Business Component dialog box opens and displays the component tree.




In the component tree, the status of each component is indicated by its icon. For more information, refer to the *HP Business Process Testing User's Guide*.

**2** Select the folder in which you want to save the component. To expand the tree and view a sublevel, double-click a closed folder. To collapse a sublevel, double-click an open folder.

You can either save the component to an existing folder in your Quality Center project or click the **New Folder** button to create a new folder in which to save it. If you want to save a copy of an existing component with same name, you must save it to a different folder.

**3** In the **Component Name** box, enter a name for the component. Use a descriptive name that will help you and others identify the component easily. The component name cannot exceed 220 characters (including its path, for example, Components/CompFolder1/MyComponent), cannot contain begin or end with spaces, and cannot contain the following characters:
\ / : " ? < > | * ! { } ' % ;

4 Accept the default **Component Type—QuickTest Component**.

5 Click **OK** to save the component and close the dialog box. As QuickTest saves the component, the operations that it performs are displayed in the status bar.

The component is saved to the Quality Center project. You can now view and modify it using QuickTest.

---

**Note:** Subject Matter Experts can also access the component from the Quality Center Business Components module. For more information, refer to the *HP Business Process Testing User's Guide*.

---

**Tip:** If the component was saved previously, you can save it by choosing **File** > **Save**, or clicking **Save**.

---

## Working with Manual Components

In QuickTest, you can convert a manual component created in Quality Center to an automated business component. You can then view, modify, debug, or run it in the same way as any other business component.

After you convert a manual component to an automated business component, you can still view its manual steps in Quality Center, and you can run it as a manual component using the Quality Center Manual Runner. In Quality Center, you can also modify or add additional steps to an automated component in the Automation tab. These steps are then updated automatically in the Design Steps tab, and in QuickTest.

**Note:** You can also convert a manual component to an automated component from within Quality Center. For more information, refer to the *HP Business Process Testing User's Guide*.

## Opening and Converting Manual Components

In QuickTest, you can open a manual component stored in your Quality Center project and convert it to an automated business (keyword-driven) component. When you open a manual component, QuickTest asks whether you want to convert it to a business component.

**Note:** Components that are currently open in Quality Center or another QuickTest session are locked and can be opened only in read-only format. To work with these components, they must be closed everywhere else.

**To open and convert a manual component:**

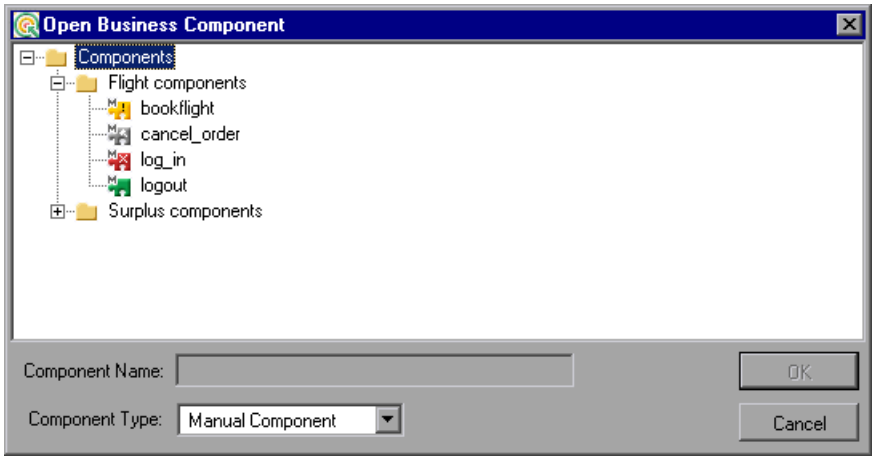
**1** In QuickTest, connect to the Quality Center project in which your component is saved. For information on connecting to Quality Center, see "Connecting to Your Quality Center Project" on page 44.

**2** Perform one of the following:

- Choose **File** > **Open** > **Business/Scripted Component**.
- Click the **Open** down arrow and choose **Business/Scripted Component**.

The Open Business Component dialog box opens showing the components stored in the Quality Center project.

You can change the type of components displayed in the dialog box, as described in step 3.




In the component tree, the status of each component is indicated by its icon. For more information, refer to the *HP Business Process Testing User's Guide*.

**3** If required, filter the list of components shown in the Business Component dialog box by selecting the component type you want to open from the **Component Type** list. By default, only QuickTest components are displayed. (QuickTest components are components that were automated using QuickTest Professional or the Business Process Testing Keyword View.)

Select one of the following component types:

- **Manual Component:** Shows components that were created in Quality Center and have not yet been converted to automated components. If you choose to open a manual component, it is converted to a QuickTest component and its manual steps are converted to Keyword View steps. Note that this conversion process is irreversible. (Although you can still view and run the manual steps in Quality Center, if needed.)
- **All Components:** Shows all QuickTest automated components and manual components.

4 Click the relevant folder in the component tree. To expand the tree and view the business components, double-click closed folders. To collapse the tree, double-click open folders.

5 Select a manual component. Manual components are represented by a component icon with an **M** in the left corner of the icon. The component name is displayed in the read-only **Component Name** box.

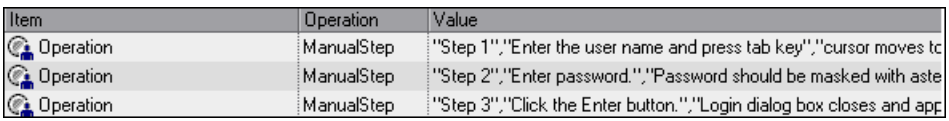
6 Click **OK** to open the component. QuickTest asks whether you want to convert the manual component to a business component.

7 Click **Yes** to continue with the conversion. Note that this process is irreversible.

8 The New Business Component dialog box opens, in which you choose an application area for your business component. Select an application area and click **OK**. For more information on application areas, see Chapter 12, "Working with Application Areas".

As QuickTest downloads, opens, and converts the component, the operations it performs are displayed in the status bar.

Each manual step from the manual component is converted into a ManualStep operation in the Keyword View.



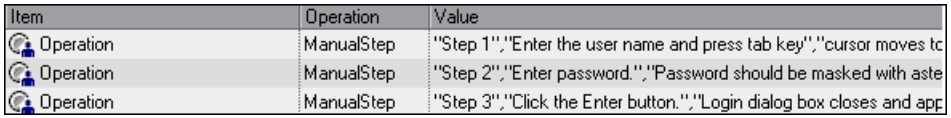
| Item | Operation | Value |
|---|---|---|
| Operation | ManualStep | "Step 1","Enter the user name and press tab key","cursor moves to |
| Operation | ManualStep | "Step 2","Enter password.","Password should be masked with aste |
| Operation | ManualStep | "Step 3","Click the Enter button.","Login dialog box closes and app |

The name, description, and expected result of each manual step are added as argument values for each ManualStep operation. Any defined input and output parameters are converted into local parameters.

You can now work with the component like any other component. You can also add additional manual steps, and modifying existing manual steps, so that you can run your business component as a manual component using the Manual Runner in Quality Center. For more information, see "Adding and Modifying Manual Steps for Components" on page 471.

## Adding and Modifying Manual Steps for Components

When you convert a manual component to an automated component, each manual step from the manual component is converted into a ManualStep operation in the Keyword View.



| Item | Operation | Value |
|---|---|---|
| Operation | ManualStep | "Step 1","Enter the user name and press tab key","cursor moves to |
| Operation | ManualStep | "Step 2","Enter password.","Password should be masked with aste |
| Operation | ManualStep | "Step 3","Click the Enter button.","Login dialog box closes and app |

You can modify step names, step descriptions, and expected results by changing the corresponding argument values in the relevant ManualStep row of the Keyword View.

You can add new steps to the converted component in QuickTest (regular business component steps and also ManualStep operations). You can also add keyword-driven steps in the Automation tab in Quality Center. You can also delete steps as needed.

All modifications you make in QuickTest to the component's ManualStep operations and regular keyword-driven steps are reflected in the Design Steps tab and Automation tab of the component in Quality Center and vice versa (after you save the changes). This means that you can update components in either Quality Center or QuickTest and still continue to run them manually using the Quality Center Manual Runner when needed.

For general information on adding steps in the Keyword View, see "Working with the Keyword View" on page 507. For more information on the ManualStep operation, refer to the **Utility** section of the *QuickTest Professional Object Model Reference*.

For more information on adding steps in Quality Center, and on running manual components using the Manual Runner in Quality Center, refer to the *HP Business Process Testing User's Guide*.

# Changing the Application Area Associated with a Component

When you create a business component in QuickTest, 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 change the application area associated with a specific component. For example, the object repository could have been modified or removed from the application area. Alternatively, 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 change the application area associated with the component.

---

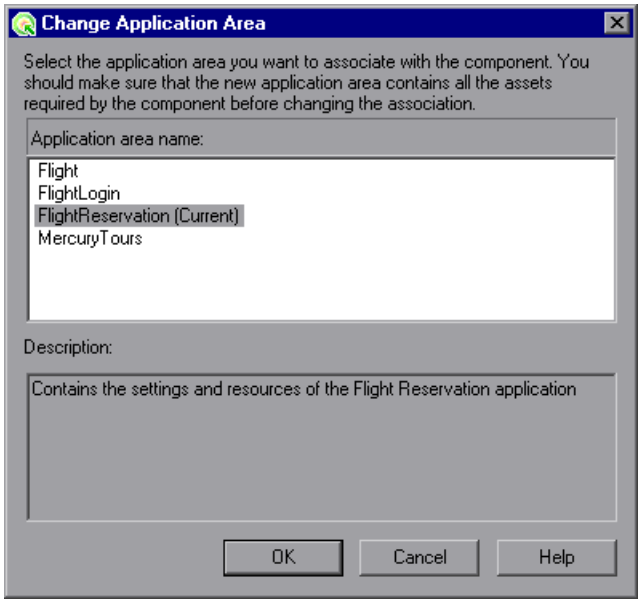
**Note:** Each time you open a component, QuickTest verifies that the resources specified for the component are available. If a component or application area has resources that cannot be found, such as a missing shared object repository, QuickTest indicates this in the Missing Resources pane. For more information, see Chapter 28, "Handling Missing Resources."

---

**To change the application area:**

1 Open the component as described in "Opening a Business Component" on page 461.

2 Choose **File** > **Change Application Area**. The Change Application Area dialog box opens.




3 Select the application area you want to associate with the component. A description of the application area is displayed in the **Description** area.

4 Click **OK** to change the application area associated with the component.

# Printing a Component

You can print your component in table format.

**To print a component:**

1. Click the **Print** button or choose **File** > **Print**. A standard Print dialog box opens.
2. Click **OK** to print the content of the Keyword View to your default Windows printer.

---

**Tip:** You can choose **File** > **Print Preview** to display the Keyword View on screen as it will look when printed.

---

# 14

# Creating Scripted Components

Scripted components are maintainable, reusable scripts that perform a specific task. Scripted components share functionality with both test actions and business components. You can use the Keyword View, the Expert View, and other QuickTest tools and options to create, view, modify, and debug scripted components in QuickTest. You can also convert existing business components or existing actions into scripted components.

**This chapter includes:**

# About Scripted Components

You can utilize the full power of both the Keyword View and the Expert View, as well as other QuickTest tools and options, when working with 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. Using the Expert View, you can enhance the scripted component flow by manually entering standard VBScript statements and other programming statements using QuickTest objects and methods. You can also incorporate user-defined functions in your scripted component steps, parameterize selected items, and add checkpoints and output values to your scripted component.

You can create scripted components for Subject Matter Experts, for example, if they need components that contain more complex functionality, such as loops or conditional statements. Subject Matter Experts working in Quality Center can then include these scripted components in business process tests to check that the application behaves as expected.

After you create a scripted component, Subject Matter Experts can view the auto-documentation generated by the component (read-only) in the Business Components module of the Quality Center project. They can run the scripted component and add it to their business process tests, but you remain responsible for maintaining the scripted component in QuickTest, if any changes are needed. Scripted components cannot be modified in Quality Center.

You save and open scripted components in the same way as you save and open business components. For more information, see “Saving a Business Component” on page 464 and “Opening a Business Component” on page 461.

### Similarities Between Scripted Components and Other Testing Documents

Scripted components contain much of the same functionality as QuickTest actions and tests. For example, you can:

- Work with programmatic statements in the Expert View (see Chapter 9, "Working in Function Library Windows.")
- Create checkpoints and output values.
- View the hierarchical Keyword View display.
- Create and work with virtual objects.
- Use the Data Table to run multiple iterations.
- 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.

For general information on all of the functionality available for scripted components, refer to the *HP QuickTest Professional User's Guide*.

Scripted components are also similar to business components, in that scripted components are:

- Associated with a specific application area. Note that all of the resources must be stored in the Quality Center project and not the file system.
- Standalone modular units that can be incorporated in a business process test.
- Linear within a business process test (not hierarchical).
- Not nested, meaning they cannot call another component.

# Creating a Scripted Component

When QuickTest is connected to a Quality Center project, you can create a new scripted component in that project.

Each scripted component is based on a specific application area, which contains the resources and settings used by the component, such as the location of the shared object repository and function libraries. You select the application area that is best suited for your scripted component. You can choose from any application area that is located in the Quality Center project in which you intend to save the component. For more information, see "Working with Application Areas" on page 413.

---

**Tip:** You can also convert a business component to a scripted component. For more information, see "Converting a Business Component to a Scripted Component" on page 482.

---

---

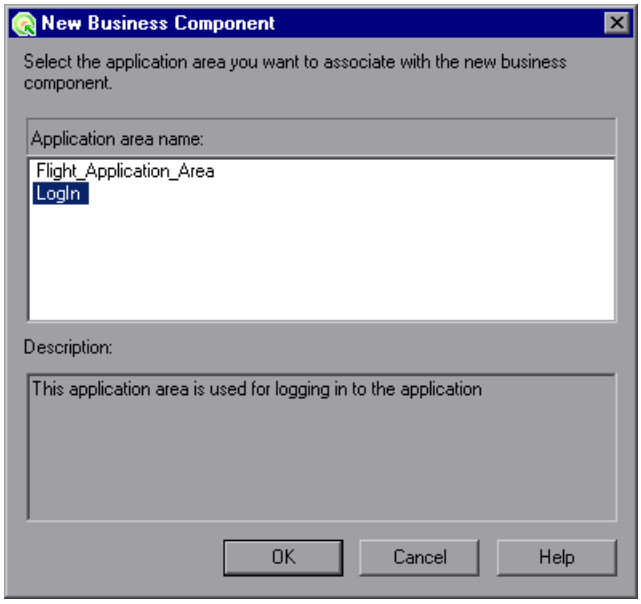
**Note:** If you have not yet created an application area, the scripted component will be based on the default application area settings provided with Business Process Testing.

---

**To create a scripted component:**

1. Connect to the Quality Center project in which you want to save the scripted component. For more information, see "Connecting to Your Quality Center Project" on page 44.
2. Perform one or the following:
   - ➤ Choose **File** > **New** > **Scripted Component**.
   - ➤ Click the **New** button down arrow and choose **Scripted Component**.

The New Business Component dialog box opens, listing all available application areas.




---

**Note:** If you have not yet created an application area, a new, untitled scripted component opens using the default settings that are supplied with Business Process Testing.

---

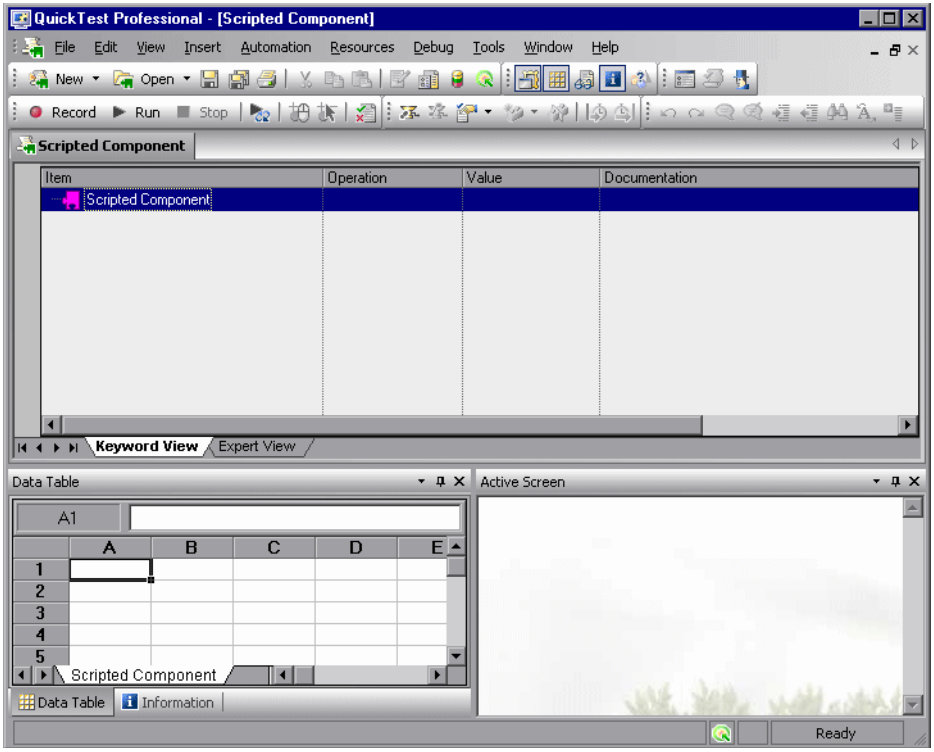
---

**Tip:** If a scripted component is already open, you can also click the **New** toolbar button to open a new scripted component.

---

**3** Select a suitable application area from the **Application Area** box. For example, if you want to create a scripted component for a Flight Reservation module, select the application area that is defined for it. Click **OK**.

A new, untitled scripted component opens. Although the scripted 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.



**4** You can now:

- Add content to your scripted component using the functionality and options provided by QuickTest. For example, in the Expert View, you can manually enter standard VBScript statements, as well as add statements using QuickTest objects and methods. You can use the Step Generator to add steps containing programming logic. You can also add checkpoints and output values to your scripted component. For more information on the functionality that can be used when creating a scripted component, refer to the *HP QuickTest Professional User's Guide*.
- Save your scripted component. (You can add content later.) You save a scripted component in the same way as you save a business component. For more information, see "Saving a Business Component" on page 464.

---

**Note:** In the Design Steps tab of the Quality Center Business Components module, Subject Matter Experts can view and work with only 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 QuickTest by clicking the **Launch** button in the Automation tab (provided that QuickTest is installed on the Quality Center client). For more information, refer to the *HP Business Process Testing User's Guide*.

---

## Converting to Scripted Components

You can convert business components and actions to scripted components, when required. When using Business Process Testing, it is generally preferable to create new business components in Quality Center rather than convert existing test actions or business components to scripted components, as this enables Subject Matter Experts working in Quality Center to maintain the components over time. In addition, because scripted components can be modified only in QuickTest (and not in Quality Center), Subject Matter Experts cannot view the automated steps in Quality Center, although they can view and modify the manual steps, if any. The conversion process is not reversible, meaning you cannot convert the scripted component back to an action or business component.

## Converting a Business Component to a Scripted Component

You can convert a single business component to a scripted component.

**To convert a business component to a scripted component:**

1 Open the business component you want to convert to a scripted component. For information on opening a business component, see "Opening a Business Component" on page 461.

---

**Note:** A business component cannot be converted to a scripted component if it is opened in read-only mode, or if it is locked.

---

2 Choose **File** > **Convert to Scripted Component**.

3 When prompted, click **OK** to proceed with the conversion.

---

**Note:** This operation replaces the existing business component with a scripted component, and cannot be undone.

---

After the conversion is complete, QuickTest automatically opens the new scripted component.

## Converting an Action to a Scripted Component

You use the Action Conversion Tool (**Start** > **QuickTest Professional** > **Tools** > **Action Conversion Tool**) to convert test actions that were created using QuickTest Professional to scripted components for use in Business Process Testing.

The Action Conversion Tool can convert only actions within tests that are saved in a Quality Center server with Business Process Testing support (license).

Every time you open the Action Conversion Tool, you connect to one Quality Center project. If you want to convert actions in multiple projects, you can close the Action Conversion Tool, reopen it, and connect to a different project.

Before you begin using the Action Conversion Tool, make sure you have the permissions required to add a component, and, optionally, a component folder, to your Quality Center project.

For more information, see:


- "Understanding the QuickTest Professional Action Conversion Tool" on page 483
- "QuickTest Professional Action Conversion Tool at a Glance" on page 485
- "Understanding Conversion Requirements" on page 488
- "Adding Actions to the Action Conversion Tool" on page 490
- "Where Do You Go From Here" on page 493
- "Modifying Actions Prior to Conversion" on page 494
- "Specifying a Name and Location for Your Scripted Component" on page 498
- "Converting Actions to Scripted Components" on page 500
- "Understanding the Conversion Logs" on page 504


## Understanding the QuickTest Professional Action Conversion Tool

The Action Conversion Tool can help you convert your existing actions to scripted components. A scripted component is a reusable script that performs a specific task in your application. Scripted components can be combined with business components to build business process tests in your Quality Center project (with Business Process Testing support).

When using Business Process Testing, it is generally preferable to create new business components in Quality Center rather than convert existing test actions to scripted components, as this enables Subject Matter Experts working in Quality Center to maintain the components over time. In addition, scripted components can be modified only in QuickTest (and not in Quality Center). Therefore, Subject Matter Experts cannot modify scripted components, although they can view scripted components in read-only mode.

You may need to perform additional tasks before conversion to enable the scripted component to run correctly. For example, after you add an action to the Action Conversion Tool, you may need to modify the action to make it suitable for conversion. For more information, see "Understanding Conversion Requirements" on page 488 and "Modifying Actions Prior to Conversion" on page 494. After you convert test actions to scripted components, you can plan and create your business process test.

The conversion process is only one stage in preparing your test actions for use in Business Process Testing. After you convert an action to a scripted component, the resources that were associated with the action (for example, shared object repositories, function libraries, recovery scenarios, and QuickTest add-ins) are no longer associated with the converted scripted component. Therefore, after the conversion process, you must open each scripted component in QuickTest and associate it with the application area that can provide the component with the required resources and settings.

When you convert an action to a scripted component, you perform the following steps:

➤ Choose a test or a test folder that contains the actions you want to convert and add it to the Action Conversion Tool. When you add a test or a test folder, the Action Conversion Tool analyzes the test(s) to determine if the actions can be converted, or not. If an action cannot be converted, the reason is listed in the **Status** box. For more information, see "Understanding Conversion Requirements" on page 488.

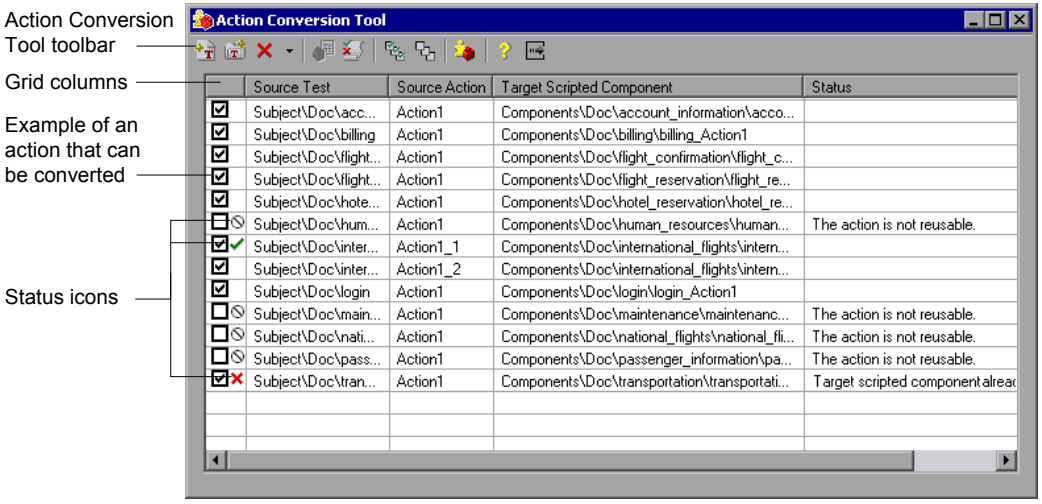
You can generate a log file to view a list of all actions that can and cannot be converted, together with an explanation of why a particular action cannot be converted.

- Select the check box(es) for the actions that you want to convert. You can modify the name and path of the target scripted component, if needed.
- Convert the action(s) to scripted components.
- View the conversion results.

After you convert an action, you can view it in both QuickTest and your Quality Center project. Before you begin working with the component, though, you need to associate it with application that can provide it with all of the required resources and setting.

## QuickTest Professional Action Conversion Tool at a Glance

The following illustrates the different parts of the Action Conversion Tool.



The Action Conversion Tool window contains the following key elements:

### Grid Columns

The Action Conversion Tool grid is divided into several columns. Each column shows different information.

- **Status icon.** Displays the current status of an action. Possible statuses are:
  ✔ indicates that the action was converted successfully.
  ✖ indicates that the conversion failed.
  ⃠ indicates that the action cannot be converted.

  In addition, this column displays whether an action is selected or not:
  ☑ indicates that the action is selected and can be converted.
  ☐ indicates that the action is not currently selected and/or cannot be converted.
- **Source Test.** Indicates the path of the test containing the action to be converted. For more information, see "Adding Actions to the Action Conversion Tool" on page 490.
- **Source Action.** Indicates the action to be converted. For more information, see "Adding Actions to the Action Conversion Tool" on page 490.
- **Target Scripted Component.** Indicates the path and location in Quality Center in which the converted action is to be saved as a scripted component. When you add an action, the Action Conversion Tool provides a default name and path for it that you can modify, if required. For more information, see "Specifying a Name and Location for Your Scripted Component" on page 498.
- **Status.** Indicates the current status of the action. If the **Status** box is empty, the action either can be or is already converted to a scripted component. If the **Status** box contains text, the message indicates why the action cannot be converted to a scripted component. For more information, see "Understanding the Conversion Logs" on page 504.

## Toolbar

The Action Conversion Tool toolbar contains buttons to assist you in converting your test actions to scripted components.

The following table lists the buttons in the Action Conversion Tool toolbar.

| Button | Name | Description |
|---|---|---|
|  | **Add Test** | Adds an individual test to the Action Conversion Tool window. |
|  | **Add Test Folder** | Adds all of the tests in the specified test folder to the Action Conversion Tool window. |
|  | **Remove From List** | Removes the highlighted action(s) from the list. Clicking the down arrow provides the following removal options:<br>➤ **Remove Highlighted.** Removes the actions that are highlighted in the list.<br>➤ **Remove All.** Removes all actions from the list.<br>➤ **Remove Converted.** Removes all actions that are already converted from the list.<br>➤ **Remove Non-Convertible.** Removes from the list any action that cannot be converted.<br>➤ **Remove Failed.** Removes from the list all actions that could not be successfully converted. |
|  | **Edit Scripted Component Properties** | Opens the Edit Scripted Component Properties dialog box, enabling you to modify the path to which the scripted file will be saved. |
|  | **Generate Convertible Status Log** | Generates a text file that specifies which actions are convertible and which are not. Also specifies the reason(s) why an action cannot be converted. |
|  | **Check All** | Selects all of the actions in the window that are suitable for conversion. |
|  | **Uncheck All** | Clears all of the actions in the window so that none are selected. |

| Button | Name | Description |
|---|---|---|
|  | **Convert Checked Actions** | Converts the selected actions to scripted components and saves them to your Quality Center project, as specified in the Target Scripted Component column. |
|  | **Help** | Opens the online help for the Action Conversion Tool. |
|  | **Exit** | Closes the Action Conversion Tool window. |

## Understanding Conversion Requirements

You can use the Action Conversion Tool to convert actions that are suitable for use as scripted components. These include actions that comply with the requirements of Business Process Testing, as described in the *HP Business Process Testing User's Guide* (part of the Quality Center documentation set).

Some actions may need to be modified before they can be converted to scripted components because QuickTest tests are designed and built differently than business components or scripted components. If an action requires extensive modification to convert it to a scripted component, you may decide that it preferable to create a new business component, which can be maintained in your Quality Center project.

---

**Note:** For information on modifying tests and actions in QuickTest, refer to the *HP QuickTest Professional User's Guide*.

---

You can use the following guidelines to determine whether an action is suitable for conversion and to decide what preparatory steps you need to perform prior to converting an action to a scripted component. Only actions that meet these guidelines are suitable for conversion to scripted components.

- The action name must contain only English characters.
- The action must be reusable.
- The action cannot call another action (a nested action).
- Action parameters cannot be defined with the value type, **Any** (in the Parameters tab of the Action Properties dialog box).
- If the action uses a shared object repository, that repository must be saved in the Quality Center project to which you are connected.
- If the action accesses external resource files, such as a function library file, a recovery scenario file, and so forth, the files must be saved in your Quality Center project.
- The action cannot refer to a global data sheet in the Data Table.
- If the test containing the action was created using an earlier version of QuickTest, the test must be upgraded to the current version of QuickTest before the action can be converted to a component. (You upgrade the test by opening and saving the test using the current version of QuickTest.)

---

**Note:** The Action Conversion Tool may not recognize that an action refers to the global data sheet in the following cases:

- If the global data sheet was renamed to something other than **Global**.
- If the action previously contained at least one local parameter, which was changed to a global parameter (whose value is stored in the global data sheet).

In these cases, the Action Conversion Tool may convert the action, but when the scripted component runs, it will fail. (This is because components can only access local data sheets.)

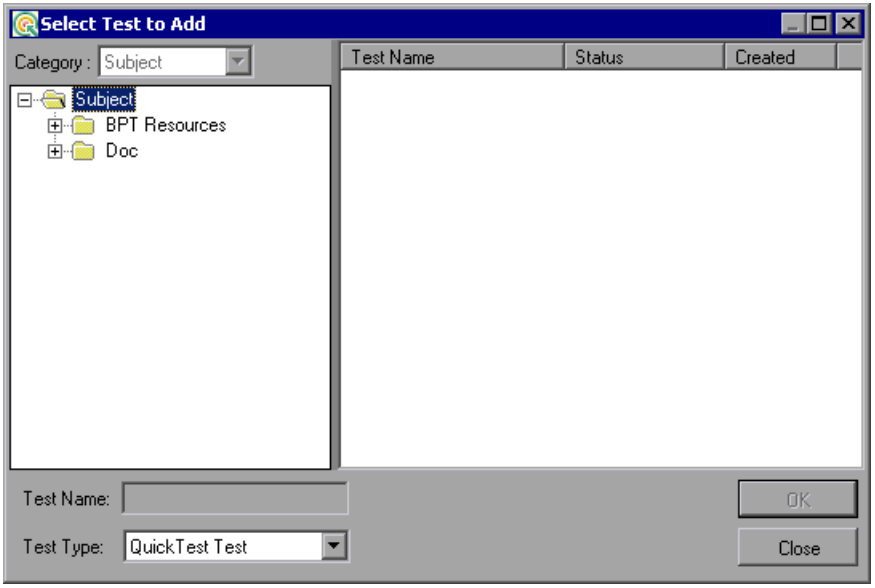
---

## Adding Actions to the Action Conversion Tool

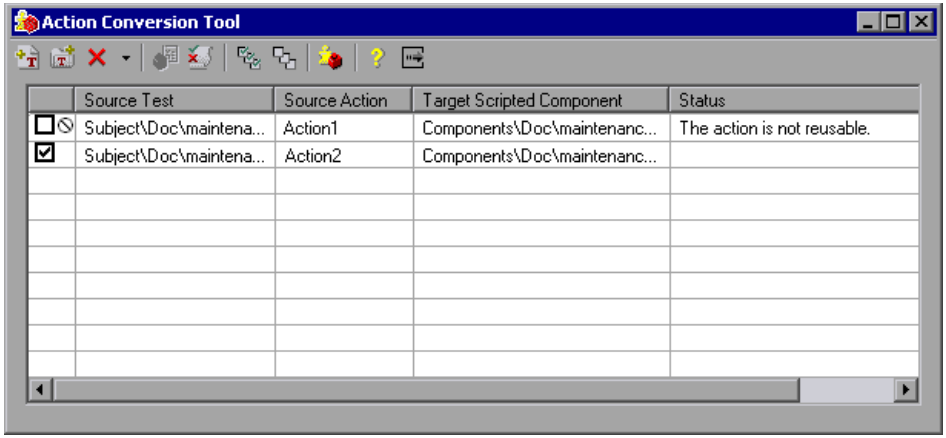
You add actions to the Action Conversion Tool by adding individual tests or test folders that contain the actions that you want to convert. You can choose to add one test at a time or to add all tests located in a specific folder. All tests must be located in your Quality Center project.

**To add a single test to the Action Conversion Tool:**

**1** Click the **Add Test** button. The Select Test to Add dialog box opens.



**2** Browse to the test that you want to add and click **OK**. The Action Conversion Tool analyzes the test to determine whether its actions can be converted. It then adds the actions in the test to the Action Conversion Tool window and displays each action and its details in a separate row in the grid.




The **Status** column shows the convertible status of each action:

- If the **Status** box is empty, the action is suitable for conversion.
- If the **Status** box contains a message, the action is not currently suitable for conversion and the reason for this is displayed. You may need to modify the action in QuickTest and re-add it to the Action Conversion Tool before you can convert it to a scripted component. For more information about whether an action can be converted, see "Understanding Conversion Requirements" on page 488.

---

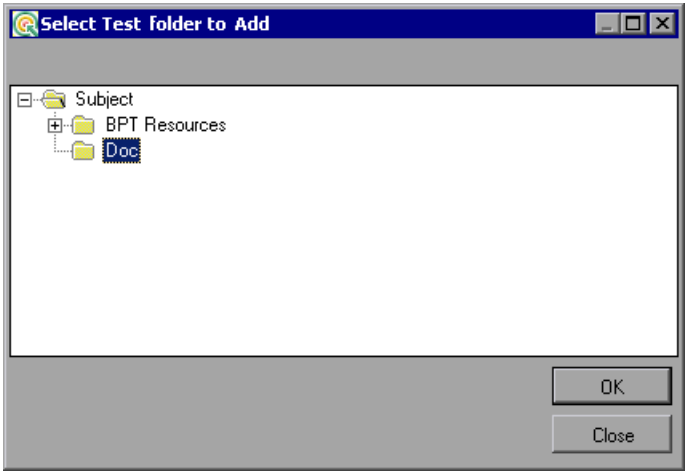
**Note:** Actions are sorted alphabetically according to the test path in the **Source Test** column. If a test contains more than one action, the actions are then sorted according to the action name in the **Source Action** column.

---

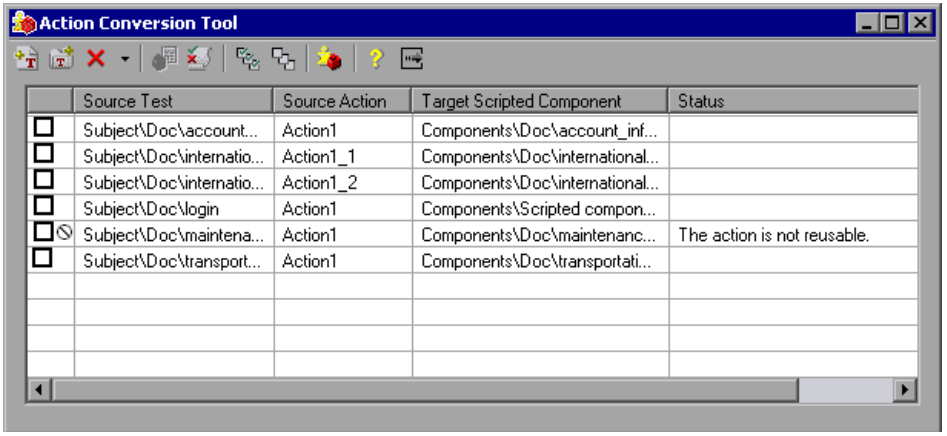
You can now perform a number of different operations. For more information, see "Where Do You Go From Here" on page 493.

**To add all of the tests in a folder to the Action Conversion Tool:**

**1** Click the **Add Test Folder** button. The Select Test Folder to Add dialog box opens.




**2** Browse to the test folder that you want to add and click **OK**. The Action Conversion Tool analyzes all of the tests in the folder to determine whether their actions can be converted. It then adds all of the actions in the tests to the Action Conversion Tool window and displays each action and its details in a separate row in the grid.



The **Status** column shows the convertible status of each action:

- If the **Status** box is empty, the action is suitable for conversion.
- If the **Status** box contains a message, the action is not currently suitable for conversion and the reason for this is displayed. You may need to modify the action in QuickTest and re-add it to the Action Conversion Tool before you can convert it to a scripted component. For more information about whether an action can be converted, see "Understanding Conversion Requirements" on page 488.

---

**Note:** Actions are sorted alphabetically according to the test path in the **Source Test** column. If a test contains more than one action, the actions are then sorted according to the action name in the **Source Action** column.

---

You can now perform a number of different operations. For more information, see "Where Do You Go From Here" on page 493.

## Where Do You Go From Here

After you add actions to the Action Conversion Tool, you can:

- add more actions contained in tests or in test folders (see "Adding Actions to the Action Conversion Tool" on page 490).
- generate and review the Conversion Logs, which provides details about the status of the actions displayed in the Action Conversion Tool (see "Understanding the Conversion Logs" on page 504).
- modify any actions that need updating before they can be converted and then save and re-add them to the Action Conversion Tool (see "Modifying Actions Prior to Conversion" on page 494 and "Adding Actions to the Action Conversion Tool" on page 490).
- modify the name or location in the Quality Center project in which the scripted component will be stored (see "Specifying a Name and Location for Your Scripted Component" on page 498).
- convert checked actions to scripted components (see "Converting Actions to Scripted Components" on page 500).

## Modifying Actions Prior to Conversion

As you add actions to the Action Conversion Tool, they are analyzed to determine if they are suitable for conversion. If an action is not suitable for conversion, the Action Conversion Tool displays a status message (in the **Status** column of the Action Conversion Tool window and in the log files) that describes why the action is not suitable for conversion. (If an action is suitable for conversion, no status message is displayed.)

The following table suggests solutions for status messages that may be displayed for an action. After you make your changes, save the test and re-add the action to the Action Conversion Tool.

| Status Message | Solution |
|---|---|
| **The action is not reusable.** | Only reusable actions can be converted to scripted components.<br>**To make the action reusable:**<br>1 Right-click the action name and select **Action Properties** from the context menu. The Action Properties dialog box opens.<br>2 In the General tab, select the **Reusable action** check box. |
| **The action is calling another action.** | Nested actions are not supported for components.<br>Modify the action so that it does not contain calls to nested actions. |
| **The action contains a parameter of type "Any".** | The parameter value type, **Any** (defined in the Parameters tab of the Action Properties dialog box), is not supported for components.<br>Select another value type for the action parameter or remove the parameter from the action. |

| Status Message | Solution |
|---|---|
| **The action uses a parameter in a global data sheet.** | If the action uses a parameterized checkpoint, the value must be stored in a local data sheet.<br>**To modify the checkpoint to use a local (current action) data sheet:**<br>1 Open the Checkpoint Properties dialog box for the parameterized checkpoint and click the **Parameter Options** button. The Parameter Options dialog box opens.<br>2 Select the **Current action sheet (local)** radio button and click **OK**. The data sheet column from the global data sheet is automatically copied to the local data sheet. Note that only the data in the first row is used by the scripted component.<br>3 Click **OK** to close the Checkpoint Properties dialog box. |
| **The action's script refers to global data sheet.** | Components cannot use global data sheets. (Information cannot be passed between actions.)<br>Modify the test so that it uses a local (per-action) data sheet (and not a global data sheet). |
| **The action uses a shared object repository that is not saved in Quality Center.** | Components can use only shared object repository files that are stored in the Quality Center project.<br>Save the shared object repository as an attachment in the Test Plan module of your Quality Center project (for example, Subject/BPT Resources/Object Repositories). Then, in the source test, associate the file with the action using the Associate Repositories dialog box (**Resources** > **Object Repository** > **Tools** > **Associate Repositories**).<br>After you convert the action to a scripted component, you need to associate the component with the application area that provides access to this shared object repository. For more information, see “Converting Actions to Scripted Components” on page 500. |

| Status Message | Solution |
|---|---|
| **The source test uses a function library that is not saved in Quality Center.** | Components can use only function library files that are stored in the Quality Center project.<br>Save the function library file as an attachment in the Test Plan module of your Quality Center project (for example, Subject/BPT Resources/Libraries). Then, in the source test, associate the file with the test (using the Test Settings dialog box).<br>After you convert the action to a scripted component, you need to associate the component with the application area that provides access to this function library. For more information, see "Converting Actions to Scripted Components" on page 500. |
| **The source test uses an environment file that is not saved in Quality Center.** | Components can use only environment files that are stored in the Quality Center project.<br>Save the environment file (as an attachment in **.xml** format) to **Subject/<folder name>** in the Test Plan module of your Quality Center project. Then, in the source test, associate the file with the test (using the Test Settings dialog box). |
| **The source test uses a recovery scenario file that is not saved in Quality Center.** | Components can use only recovery scenario files that are stored in the Quality Center project.<br>Save the recovery scenario file(s) in the Test Plan module of your Quality Center project (for example, Subject/BPT Resources/Recovery Scenarios). Then, in the source test, associate the file with the test (using the Test Settings dialog box).<br>After you convert the action to a scripted component, you need to associate the component with the application area that provides access to this recovery scenario file. For more information, see "Converting Actions to Scripted Components" on page 500. |

| Status Message | Solution |
|---|---|
| **The source test uses a Data Table that is not saved in Quality Center.** | Components can use only external Data Tables that are stored in the Quality Center project.<br>Save the external Data Table (as an attachment in **.xls** format) in the Test Plan module of your Quality Center project (for example, Subject/BPT Resources/Recovery Scenarios). Then, in the source test, associate the file with the test (using the Test Settings dialog box). |
| **This test was created using an earlier version of QuickTest.** | The Action Conversion Tool can only convert actions from tests created using the current version of QuickTest.<br>Before you can convert an action from this test, you must open and save the test using the current version of QuickTest. This upgrades the test to the current version of QuickTest. Alternatively, if you do not want to upgrade the test to the current version, you can use an earlier version of the Action Conversion Tool (installed with an earlier version of QuickTest). |

---

**Note:** The Action Conversion Tool may not recognize that an action refers to the global data sheet in the following cases:

- If the global data sheet was renamed to something other than **Global**.
- If the action previously contained at least one local parameter, which was changed to a global parameter (whose value is stored in the global data sheet).

In these cases, the Action Conversion Tool may convert the action, but when the scripted component runs, it will fail. (This is because components can only access local data sheets.)

---

## Specifying a Name and Location for Your Scripted Component

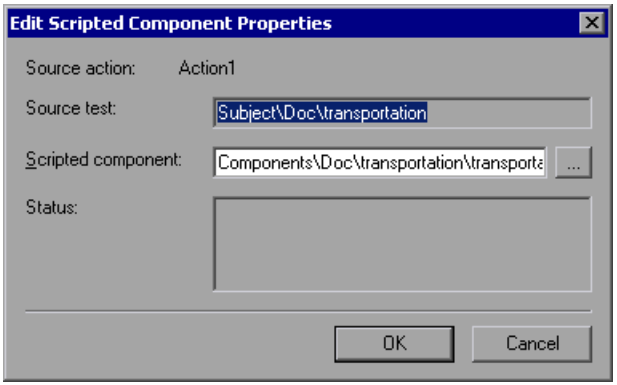
All scripted components are stored in your Quality Center project. You can specify the name and location for each component, or you can accept the default name and location suggested by the Action Conversion Tool, which has the following naming convention: **Components/<Folder name>/<Test name>/<Test name>_<Action name>**. If you need to create a new folder in which to store the scripted component, make sure you have the required Business Process Testing permissions in your Quality Center project.

**To modify the name and location of a scripted component:**

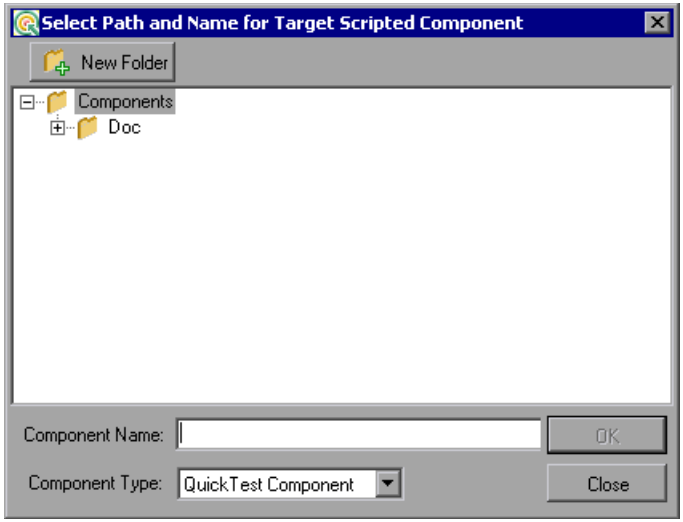
**1** Open the Edit Scripted Component Properties dialog box in one of the following ways:

- Select the row containing the target location for the scripted component and click the **Edit Scripted Component Properties** button.
- Double-click anywhere in the row containing the target location for the scripted component.

The Edit Scripted Component Properties dialog box opens.



**2** Modify the path in the **Scripted component** box manually or click the browse button. If you click the browse button, the Select Path and Name for Target Scripted Component dialog box opens, enabling you to select a path and, in the **Component Name** box, enter a name.




**3** Click **OK**. The target path for the scripted component is updated.

---

**Tip:** In the Action Conversion Tool, you can also click the **Target Scripted Component** cell and modify the path manually or by using the browse button.

---

## Converting Actions to Scripted Components

When you add actions to the Action Conversion Tool, the **Status** box indicates whether or not each action can be converted. An empty **Status** box indicates that the test can be converted. If the **Status** box displays a message, then the action cannot currently be converted. Read the message and modify the action accordingly. For example, if the message states that the action is not reusable, you can open the test in QuickTest and make the action reusable. (For information on working with QuickTest, refer to the *HP QuickTest Professional User's Guide*.)

After you modify and save the test, you can add it to the Action Conversion Tool again. When you re-add a test, the Action Conversion Tool replaces the test you added previously with the newly modified test. It also applies the same target path that you defined previously. Therefore, if you chose a different location to store the scripted component after it is converted, the modified test will now use that path.

When the Action Conversion Tool converts an action, it:

- creates a new scripted component in your Quality Center project
- copies the source action's steps and content to the scripted component
- copies the test settings for the source action's parent test to the scripted component's settings (displayed in the Business Component Settings dialog box)
- copies the source action's parameterization data
- copies the first row from the data sheet to the Data Table that is used by the scripted component

---

**Note:** You use the Test Settings dialog box (**File** > **Settings** > **Resources**) to specify that a source action use an external data sheet. You use the Value Configuration Options dialog box for a specific parameter to specify that a source action use a local (current action) data sheet.

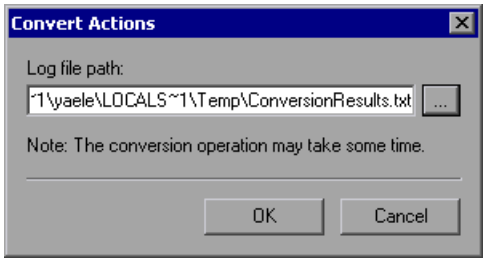
---

After you convert actions to scripted components, you must open each scripted component in QuickTest and associate it with an application area that provides all of the resources and settings needed by the component.

**To convert actions to scripted components:**

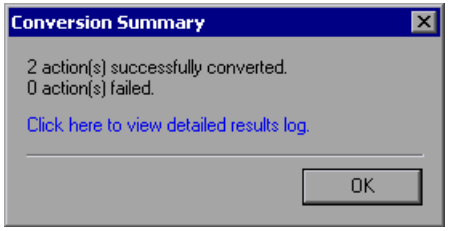
**1** Select the check boxes for the actions that you want to convert. You can click the **Check All** button to select all actions that are suitable for conversion.

**2** Click the **Convert Checked Actions** button to begin the conversion process. The Convert Actions dialog box opens.




**3** Modify the location in which the Conversion Results log file will be saved, if needed.

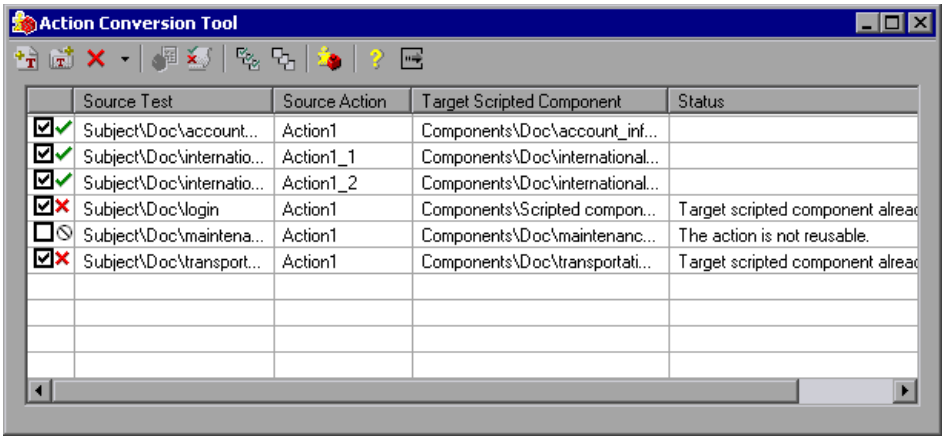
**4** Click **OK**. The Action Conversion Tool converts the selected actions to scripted components. Note that this may take some time. As the actions are converted, a progress bar is displayed. When the conversion is finished, the **Conversion Summary** box opens and displays the number of actions that were successfully converted and the number of conversions that failed.



**5** Click the **Click here to view the detailed conversion results log** link to open the conversion results log text file, or click **OK** to close the **Conversion Summary** box. If you clicked the link, a text file opens specifying:

- the number of actions that were converted successfully
- the number of actions that were not converted
- the names and details of the actions that were converted
- the names and details of the actions that were not converted and the reason(s) why they were not converted

The Action Conversion Tool also displays the status of each of the actions.



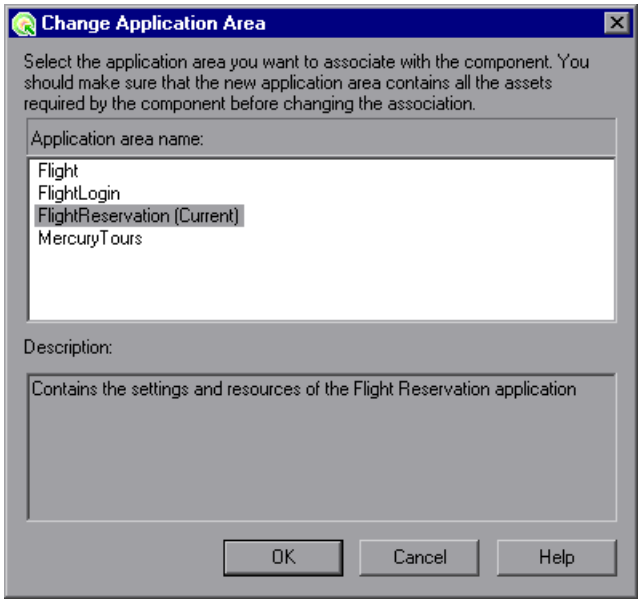

✔ next to an action indicates that the action was converted successfully.
✘ next to an action indicates that the conversion failed.
⃠ next to an action indicates that the action cannot be converted.

For more information on limitations that may appear in the **Status** column, see “Modifying Actions Prior to Conversion” on page 494.

**To associate a scripted component with an application area:**

1 In QuickTest, open the scripted component. For more information, see "Opening a Business Component" on page 461.

2 Choose **File** > **Change Application Area**. The Change Application Area dialog box opens.




3 Select a suitable application area from the **Application Area name** box. A description of the application area is displayed in the **Description** area.

---

**Note:** After you associate the application area with the component, you may need to modify the application area to ensure that it provides all of the required resources and settings. For more information, see "Working with Application Areas" on page 413.

---

4 Click **OK** to associate the application area with the scripted component.

## Understanding the Conversion Logs

When you use the Action Conversion Tool, you can generate printable log files that specify the status of each action—both before and after you convert your actions to scripted components. The following table describes the details provided by each of these conversion log files:

| Log File Name | Description |
|---|---|
| **ConvertibleStatus.txt** | Indicates the convertibility status of every action in the Action Conversion Tool, prior to conversion:<br>➤ provides a summary of how many actions can be converted and how many cannot be converted (in total).<br>➤ for each source test, specifies which action(s) can be converted and which action(s) cannot be converted.<br>➤ for each action that cannot be converted, specifies the reason(s) why the action cannot be converted, for example, the action may not be reusable. |
| **ConversionResults.txt** | Indicates whether each action was converted successfully. If an action was not converted successfully, describes the reason(s) why not. It also:<br>➤ provides a summary of how many actions were successfully converted and how many were not.<br>➤ for each source test, specifies the actions that were (or were not) converted, and lists the reason(s) why a particular action was not converted. |

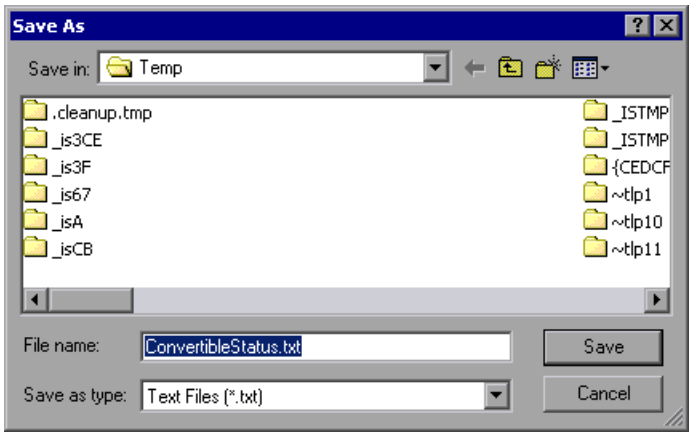
## Viewing the Convertible Status Log File

You can generate and view the convertible status log file (**ConvertibleStatus.txt**) at any time.

**To generate the convertible status log file:**

1 In the Action Conversion Tool, add the actions that you want to convert to scripted components. For more information on adding actions, see "Adding Actions to the Action Conversion Tool" on page 490.

2 Click the **Generate Convertible Status Log** button. The Save As dialog box opens.




3 Browse to the location in which you want to save the log file (you can rename it, if needed) and click **Save**. The file is saved to the specified location.

## Viewing the Conversion Results Log File

When you convert actions to scripted components, you specify the location in which you want to save the conversion results log file. After the Action Conversion Tool converts the selected actions to scripted components, it generates the **ConversionResults.txt** log file automatically. You can view the conversion results log file immediately by clicking the link. You can also view the conversion results log file at a later time by browsing to the location in which you stored it. For more information, see "Converting Actions to Scripted Components" on page 500.

# 15

# Working with the Keyword View

The Keyword View provides an easy way to create, view, and modify tests in a graphical easy-to-use format.

**This chapter includes:**

# About Working with the Keyword View

The Business Component Keyword View enables you to create and view the steps of a component in a modular, table format. Each step is a row in the Keyword View that is comprised of individual, modifiable parts. You create and modify steps by selecting items and operations in the Keyword View and entering information as required. Each step is automatically documented as you complete it, enabling you to view a description of your 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 a component and to view, modify, and debug existing component 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.

In general, the Subject Matter Expert uses the Automation tab in the Quality Center Business Components module to add content to and modify component steps. However, this can also be done in QuickTest. The Business Component Keyword View that you see in QuickTest is the same as the Automation tab that the Subject Matter Expert uses in Quality Center.

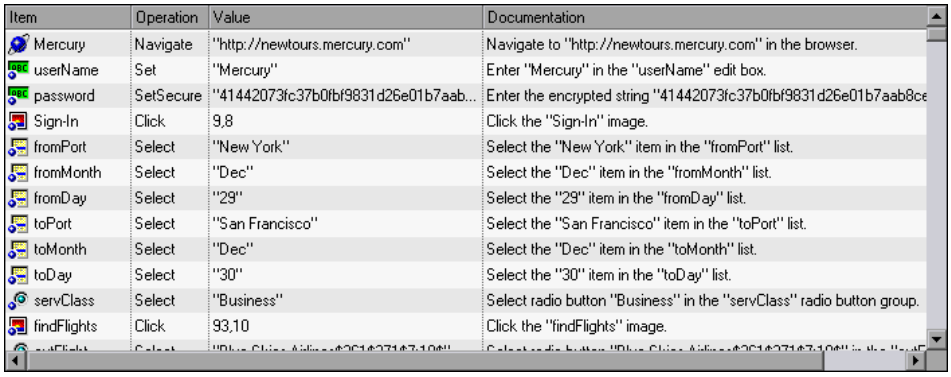
The Business Component Keyword View differs from the QuickTest Test Keyword View in that it provides component-specific options that are specially designed to facilitate the creation of business components. This makes it easy and intuitive for Subject Matter Experts to create business components in Quality Center.

The Business Component Keyword View can include comments, which enable you to enter manual steps and informational separators in a business component. All items (test objects and operations) in the Keyword View are displayed at the same hierarchical level, even if they are child objects of the previous step or operations to be performed. This makes it easier for Subject Matter Experts to manage their business component steps.

To work with the Business Component Keyword View, QuickTest must be connected to a Quality Center project with Business Process Testing support.

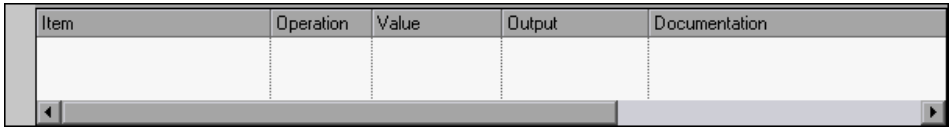
# Understanding the Keyword View

The Business Component Keyword View is comprised of a table-like view, in which each step is a separate row in the table, and each column represents the different parts of the steps. The columns displayed vary according to your selection. For more information, see “Defining Keyword View Display Options” on page 545.



| Item | Operation | Value | Documentation |
|---|---|---|---|
| Mercury | Navigate | "http://newtours.mercury.com" | Navigate to "http://newtours.mercury.com" in the browser. |
| userName | Set | "Mercury" | Enter "Mercury" in the "userName" edit box. |
| password | SetSecure | "41442073fc37b0fbf9831d26e01b7aab... | Enter the encrypted string "41442073fc37b0fbf9831d26e01b7aab8ce |
| Sign-In | Click | 9,8 | Click the "Sign-In" image. |
| fromPort | Select | "New York" | Select the "New York" item in the "fromPort" list. |
| fromMonth | Select | "Dec" | Select the "Dec" item in the "fromMonth" list. |
| fromDay | Select | "29" | Select the "29" item in the "fromDay" list. |
| toPort | Select | "San Francisco" | Select the "San Francisco" item in the "toPort" list. |
| toMonth | Select | "Dec" | Select the "Dec" item in the "toMonth" list. |
| toDay | Select | "30" | Select the "30" item in the "toDay" list. |
| servClass | Select | "Business" | Select radio button "Business" in the "servClass" radio button group. |
| findFlights | Click | 93,10 | Click the "findFlights" image. |

When you create a new business component, the Keyword View is empty, as shown below.



| Item | Operation | Value | Output | Documentation |
|---|---|---|---|---|

As you add steps to a component, each step is defined as a single row in the Keyword View. You can add a step below the currently selected step, at the end of an existing component, or at the beginning of a new component. You can also enter standard and bitmap checkpoints, output values, and comments.

**Steps.** A step represents an operation to be performed. After you create a step, you specify its contents. For example, you can choose the test object on which the step is performed, specify the operation to be performed in the step, and specify any relevant input or output values. When a business process test is run in Quality Center, the steps defined in the associated business components are performed. This section describes how to add a step to your business component.

**Comments.** A comment is a free text entry that spans an entire row. The icon indicates a comment. You use comments to define manual steps or to provide information on adjacent steps in your business component. Comments are not processed when a business process test is run. For more information, see "Working with Comments" on page 541.

You can add steps to a component manually or by recording the steps you perform on your application. Each step is inserted as a row in the Keyword View. For example, the Keyword View could contain the following rows:

| Item | Operation | Value | Output | Documentation |
|---|---|---|---|---|
| userName | Set | "Mercury" | | Enter "Mercury" in the "userName" edit box. |
| password | SetSecure | "3ee35" | | Enter the encrypted string "4129e9544fe92be2 |
| Sign-In | Click | 35,7 | | Click the "Sign-In" image. |

These rows show the following three steps that are all performed on the **Welcome: Mercury Tours** page of the Mercury Tours sample web site:

- **Mercury** is entered in the **userName** edit box.
- The encrypted string **3ee35** is entered in the **password** edit box.
- The **Sign-In** image is clicked.
- The **Documentation** column translates each of the steps into understandable sentences.

You can use the Keyword View to add steps at any point in your component. After you add steps, you can modify or delete them using standard editing commands and drag-and-drop functionality. You can print the contents of the Keyword View to your Windows default printer (and even preview the contents prior to printing). For more information, see "Printing a Component" on page 474.

In the Keyword View, you can also view checkpoint and output value properties.

The Business Component Keyword View can contain any of the following columns: **Item**, **Operation**, **Value**, **Output**, and **Documentation**. A brief description of each column is provided below.

## Item Column

The item on which you want to perform the step—either a test object or a user-defined function (**Operation**). You must select an option from the **Item** column before you can add additional content to a step. For more information, see "Selecting an Item for Your Step" on page 516.

## Operation Column

The operation to be performed on the item. This column contains a list of all available operations (methods, functions, and sub-procedures) that can be performed on the item selected in the **Item** column, for example, **Click** and **Select**. The most commonly used operation for the item selected in the **Item** column is displayed by default. For more information, see "Selecting the Operation for Your Step" on page 525. You can define additional operations for a test object using the **RegisterUserFunc** method. For more information, see "Working with User-Defined Functions and Function Libraries" on page 373.

## Value Column

The argument values for the selected operation. The **Value** cell is partitioned according to the number of arguments of the selected option in the **Operation** column. The 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 more information, see "Working with Parameters" on page 533.

**Component parameter.** A component parameter is a parameter that can be accessed by any component in your Quality Center project. For more information, see "Defining Parameters for Your Component" on page 609.

## Output Column

The parameter in which output values for the step are stored. For 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.

## Documentation Column

Read-only auto-documentation of what the step does in an easy-to-understand sentence, for example, Click the "Sign-in" image. or Select "San Francisco" in the "toPort" list. If you want to print or view only the steps, you can choose to display only this column. For example, you may want to print or view manual testing instructions.

---

**Tips:**

- You can display only the **Documentation** column of a component by right-clicking the column header row and choosing **Documentation Only** from the displayed menu.
- You can also copy the documentation by selecting **Edit** > **Copy Documentation to Clipboard**, or right-clicking the column header row and choosing **Copy Documentation to Clipboard** from the displayed menu, and then paste it into a different application, as required.

---

---

**Note:** If you do not see one or more of these columns in the Keyword View, you can use the Keyword View Options dialog box to display them. For more information, see "Defining Keyword View Display Options" on page 545.

---

## Tips for Working with the Keyword View

- You can use the left and right arrow keys to move the focus one cell to the left or right, with the following exceptions:
  - In the **Item** column, the left and right arrow keys collapse or expand the item (if possible). If not possible, the arrow keys behave as in any other column.
  - When a cell is in edit mode, for example, when modifying a value or comment, the left and right arrow keys move within the edited cell.
- When a **Value** cell is selected, press CTRL+F11 to open the Value Configuration Options dialog box.
- When the entire step is selected (by clicking to its left), use the + key (expands a specific branch), - key (collapses a specific branch), and * key (expands all branches) to expand and collapse the **Item** tree.
- When a row is selected (not a specific cell), you can type a letter to jump to the next row that starts with that letter.

---

**Note:** In addition to the above commands, you can also use QuickTest menu shortcuts. For more information, see "Performing QuickTest Commands" on page 71.

---

# Adding a Step to Your Component

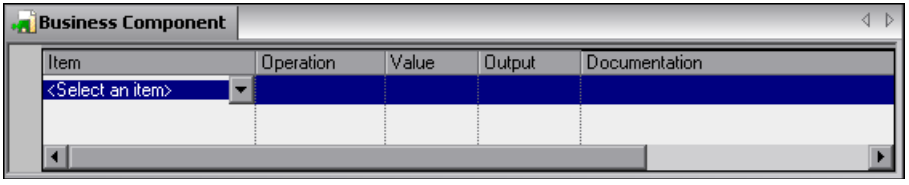
You can add a step at any point in your component. You can add a step below the currently selected step, at the end of a component, or at the beginning of a new component.

**To add a step:**

**1** Perform one of the following:

- Click anywhere in the Keyword View (below the existing steps, if any) to add a step at the end of the component. If no steps are defined yet, this adds the first step to the component.
- Choose **Insert** > **New Step** to add a new step after the existing steps (if any). If the component does not contain any steps, this adds the first step to the component.
- Select an existing step and choose **Insert** > **New Step** to add a new step between existing steps. (If you select the last step, QuickTest adds a step at the end of the component.)
- Right-click an existing step and choose **Insert New Step** from the context-sensitive menu.

A new step is added to the Keyword View below the selected step.




---

**Note:** The **Select an item** list is generally expanded to display all applicable test objects, as well as the **Operation** and **Comment** items.

---

**2** Define the step by clicking in the cell for the part of the step you want to modify and specifying its contents, as described below. Each cell in the step row represents a different part of the step. For each step, you can define the following:

- **Item.** Either a test object on which you perform a step, or a user-defined function (**Operation**). You must select an option from the **Item** column before you can add additional content to a step. For more information, see "Selecting an Item for Your Step" on page 516.

  Alternatively, you can choose to add a **Comment**, which enables you to add a manual step or other free text information between steps. For more information, see "Working with Comments" on page 541.

- **Operation.** The operation to be performed on the item. For more information, see "Selecting the Operation for Your Step" on page 525.

- **Value.** (If relevant.) The argument values for the selected operation. For more information, see "Defining Values for Your Step Arguments" on page 526.

- **Comment.** (If relevant.) Textual notes about the step. For more information, see "Working with Comments" on page 541.

- **Output.** (If relevant.) The parameter in which output values for the step are stored. For more information, see "Defining an Output Value for Your Step" on page 529.

---

**Note:** The **Documentation** cell is read-only. This cell displays an explanation of what the step does in an easy-to-understand sentence, for example, Click the "Sign-in" image. or Select "San Francisco" in the "toPort" list. In most cases, QuickTest can generate the description displayed in this cell.

If you created a function library and added (associated) it to the associated application area, QuickTest can display documentation for it only if you defined the relevant text in the function library. For more information, see "Documenting the Function" on page 399 and "Managing Function Libraries" on page 426.

---

---

**Tip:** You can use the standard editing commands (**Cut**, **Copy**, **Paste**, and **Delete**) in the **Edit** menu or in the context-sensitive 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 component. For more information, see "Managing Component Steps" on page 543 and "Using Keyboard Commands in the Keyword View" on page 544.

---

**3** After you make your changes, save the component to your Quality Center project. For more information, see "Saving a Business Component" on page 464.

## Selecting an Item for Your Step

An **item** can be a test object in the shared object repository or a user-defined function—**Operation**. (The **Operation** item is available only if user-defined functions were added to a function library that is associated with the component's application area. For more information, see "Managing Function Libraries" on page 426 and "Working with User-Defined Functions and Function Libraries" on page 373.)

This section describes:

- "Selecting a Test Object from the Item List" on page 517
- "Selecting a Test Object from the Shared Object Repository" on page 518
- "Selecting a Test Object from Your Application" on page 521
- "Selecting the Operation Item" on page 524

After you select an item, you specify an operation for it. For more information, see "Selecting the Operation for Your Step" on page 525.

**Note:** In addition to selecting an item or **Operation** in the **Item** cell, you can also select **Comment**. This instructs QuickTest to convert the selected step into a free text cell that spans the entire row. After the step is converted into a comment, it cannot be restored to a step. You use the **Comment** option to enter manual steps or to provide information on adjacent steps. For more information, see "Working with Comments" on page 541.

## Selecting a Test Object from the Item List

The test objects available in the **Item** list are the sibling and child test objects of the previous step's test object, as defined in the shared object repository. The example below shows the objects available for the step following a **userName** test object.


**To select a test object from the displayed Item list:**

1. Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
2. In the **Item** list, select the test object on which you want to perform the step. The item you select is displayed in the **Item** cell. You now need to specify an operation for the step. For more information, see "Selecting the Operation for Your Step" on page 525.

### Selecting a Test Object from the Shared Object Repository

The shared object repository includes all of the test objects that are defined in the application area on which your component is based (including those displayed in the **Item** list, above).

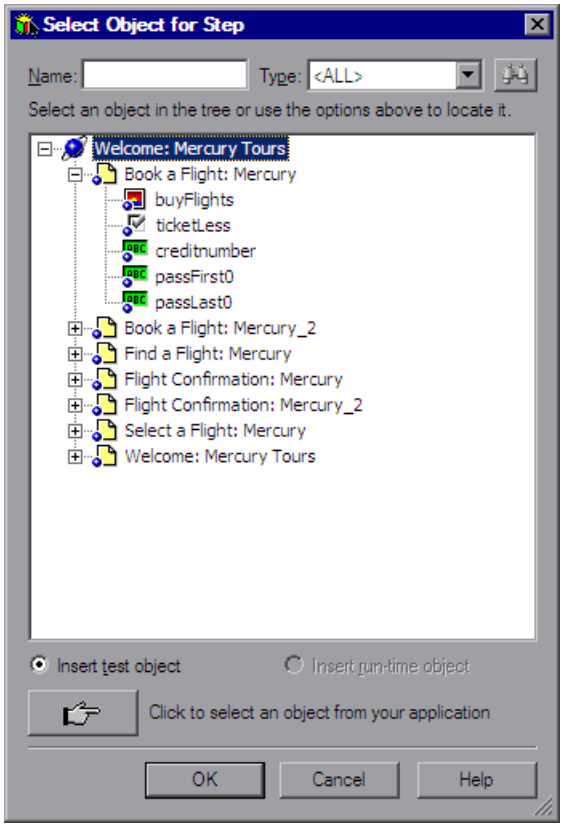
You can select any object in the object repository tree for your new step. If the object repository is very large, you can search for the object. For example, you may want to add a password object that you know is an Edit box. You can search all the **Edit** type objects for one called password, or even one containing the letter p.

For more information on the object repository, see Chapter 4, "Working with Objects." For more information on Object statements, see "Accessing Run-Time Object Properties and Methods" on page 354.

**To select a test object from the shared object repository:**

1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.

**2** In the **Item** list, choose **Select another object**. The Select Object for Step dialog box opens.




**3** Select an object from the object repository tree. If the object repository is very large, you can search for the object, as described below. If a search is not required, proceed to step8.

**4** In the **Name** box, enter the name of the object, or any part of the name. For example, you can enter p to search for all object names containing the letter p.

---

**Note:** If the **Name** box is left empty, all objects of the selected object type are considered matching criteria.

---

**5** In the **Type** box, select the type of object for which to search, or select **<All>** to search for the object in all the object types.

---

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

---

**6** Click the **Find Next** button. The search starts at the currently selected node, and the number of objects that match your criteria is displayed. The first object in the list that matches your criteria is highlighted.

**7** If required, click the **Find Next** button to navigate through all the objects that match your search criteria. The search continues to the end of the tree, then wraps to the beginning of the tree, and continues.

---

**Tip:** Press F3 to find the next object that matches your search criteria, or SHIFT+F3 to find the previous match.

---

**8** Click **OK**. The object is displayed in the **Item** column of the Keyword View. and is also added to the **Item** list. You can now specify the operation for the selected object. For more information, see “Selecting the Operation for Your Step” on page 525.

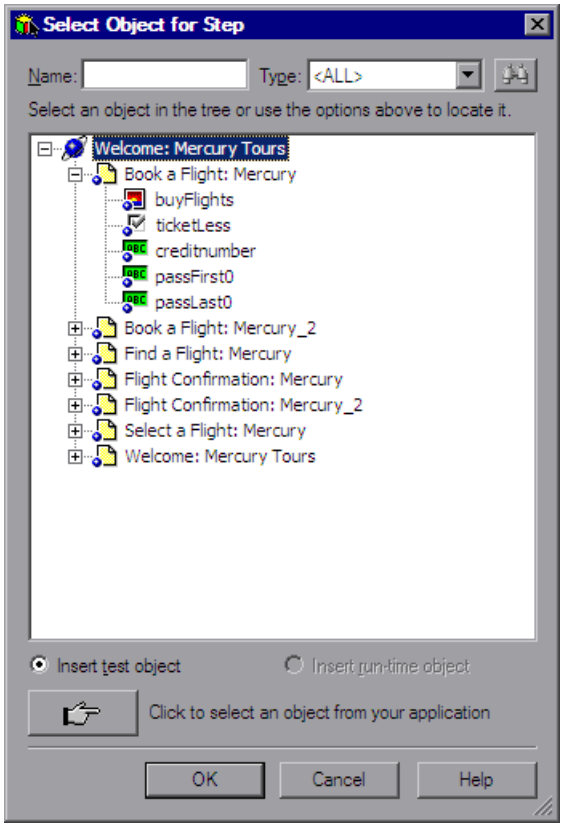
## Selecting a Test Object from Your Application

If the shared object repository does not include the test object that you need for this step, you can select it directly from your application and add it to the shared object repository so that you can use it in this and other steps.

**To add a test object from your application:**

1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.

2 In the **Item** list, choose **Object from repository**. The Select Object for Step dialog box opens.



**3** Click the pointing hand button. QuickTest is hidden.

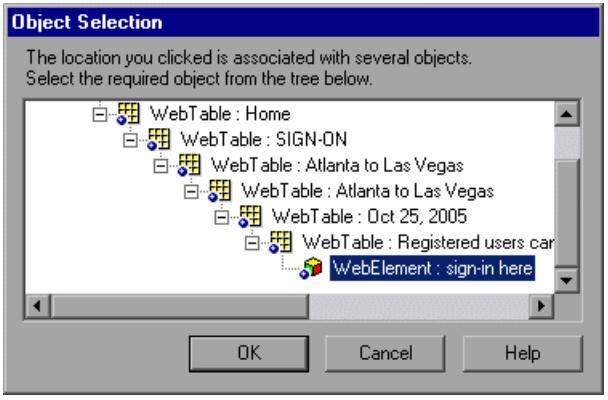
**4** Use the pointing hand to click on the required object in your application.

---

**Tip:** You can hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to click is partially hidden by another window, you can also hold the pointing hand over the partially hidden window for a few seconds until the window comes into the foreground and you can point and click on the object you want. Additionally, 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 location you clicked is associated with more than one object, the Object Selection dialog box opens.




**5** Select the object for the new step and click **OK**. The object is displayed in the shared object repository tree in the Select Object for Step dialog box.

**6** Click **OK**. The object is displayed in the **Item** column in the Keyword View. You can now specify the operation for the selected object. For more information, see "Selecting the Operation for Your Step" on page 525.

---

**Note:** Subject Matter Experts working in Quality Center can select only objects that are stored in the shared object repositories (in the component's application area).

---

---

**Tips:** If you select an object in your application that is not in the shared object repository, a test object is added to the local object repository when you insert the new step. After you add a new test object to the local object repository, it is recommended to rename it, if its name does not clearly indicate its use. For example, you may want to rename a test object named Edit (that is used for entering a username) to UserName. This will enable Subject Matter Experts to select the appropriate test object when adding steps using test objects located in this shared object repository.

After you add the required objects to the local object repository, you can use the Object Repository Merge Tool to update the shared object repository and make the new objects available to other components. For more information, see "Updating a Shared Object Repository from Local Object Repositories" on page 268.

If you are adding a container test object, it is also recommended to specify its context, for example, if you are adding a confirmation message box from a Login page, you may want to name it Login > Confirm. For more information, see "Renaming Test Objects" on page 140.

---

## Selecting the Operation Item

If your business component is based on an application area that references at least one function library, you can select the **Operation** item and select a function for the step.

User-defined functions enable you to perform a variety of additional operations, for example, open an application at the start of a business component or check the value of a specific property.

---

**Note:** If the application area on which your component is based is not associated with any function libraries, the **Operation** option does not appear in the **Item** list.

---

**To select an Operation item:**

1. Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just entered a new step, the list is displayed automatically as soon as you create the new step.
2. In the **Item** list, choose **Operation**. The **Operation** item is displayed in the **Item** cell. You now need to specify an operation for the step. For more information, see “Selecting the Operation for Your Step” on page 525.

## Selecting the Operation for Your Step

The **Operation** cell specifies the operation to be performed on the item listed in the **Item** column. The available operations vary according to the item selected in the **Item** column. When you select an item, all operations (keywords) associated with that item (via the application area) are listed.

For example, if you selected a browser test object, such as a WebButton object, the list contains all of the methods that were selected for the WebButton object from the list of available keywords in the Keywords pane of the component's application area. If you selected **Operation** in the **Item** column, the list contains the user-defined functions defined in the function library or libraries associated with the business component (via its application area).

You specify function libraries in the Function Libraries pane of the Application Area. For more information, see "Managing Function Libraries" on page 426.

**To select an operation for the step:**

Click in the **Operation** cell. Then click the down arrow button and select the operation to be performed on the item. The operation can be either a standard operation or a user-defined function. For more information on user-defined functions, see "Working with User-Defined Functions and Function Libraries" on page 373.

---

**Note:** When you position the cursor over an operation in the list, a tooltip describes what this operation performs. For user-defined functions, the tooltip is taken from the description that you provided in the associated function library. For more information, see "Documenting the Function" on page 399.

---

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.

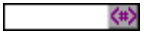
If you selected **Operation** in the **Item** column, QuickTest displays the functions that you defined in the function library, alphabetically. (You manage the function libraries for components in the Function Libraries pane of the Application Area. For more information, see "Managing Function Libraries" on page 426.)

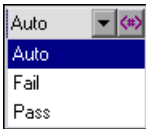
## Defining Values for Your Step Arguments

The **Value** cell lists the values for each of the operation arguments. You can insert a constant value or a parameter for each argument. If you insert a parameter, it can be either a local parameter or a component parameter. For more information, see "Working with Parameters" on page 533 and "Defining Parameters for Your Component" on page 609.

You can also encode password values. For more information, see "Inserting Encoded Passwords into Method Arguments" on page 528.

The **Value** cell is partitioned according to the number of possible arguments of the selected operation. Each partition contains different options, depending on the type of argument that can be entered in the partition, as follows:

| Argument Partition | Argument Type | Instructions |
|---|---|---|
|  | String | Enables you to enter a string containing English letters and numbers, enclosed by quotes. If you do not enter the quotes, QuickTest adds them automatically. If you modify a cell that contains a string enclosed by quotes by removing the quotes, QuickTest will not restore the quotes and the value will be treated as a variable name. |
|  | Integer | Enables you to enter any number, or use the up and down arrows to select a number. |
|  | Boolean | Enables you to select a **True** or **False** value from the list. |

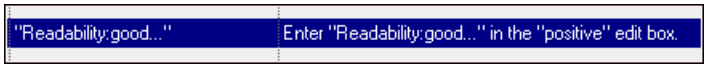
| Argument Partition | Argument Type | Instructions |
|---|---|---|
|  | Predefined Constant | Enables you to select a predefined value from the list. |

**To define or modify a value:**

Click in each partition of the **Value** cell and enter the argument values for the selected operation. Note that when you click in the **Value** cell, a tooltip displays information for each argument. In the tooltip, the argument for the partition that is currently highlighted is displayed in bold, and any optional arguments are enclosed in square brackets.


**To add multi-line arguments:**

You can also add multi-line argument values by pressing SHIFT+ENTER to add line breaks to your argument value. After you enter a multi-line argument value, QuickTest automatically converts it to a string, and displays only the first line of the argument, followed by an ellipsis (...). This format for multi-line argument values is also displayed in the Documentation column of the Keyword View.




---

**Tip:** Select the cell to display the entire argument value to be used in the step. Note that the argument value is used during the run session exactly as it appears in the step. For example, if you enter quotation marks as part of the argument value, they will be included in the argument value used during the run session. QuickTest automatically interprets a multi-line value as a string, so you do not need to add quotation marks for this purpose.

---

**To parameterize the value for an argument using a local or component parameter:**

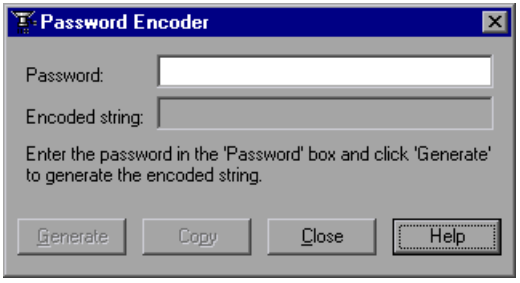
Click the  button in the required **Value** cell. For information on parameterizing a local value, see "Working with Parameters" on page 533. For information on parameterizing a component value, see "Defining Parameters for Your Component" on page 609.

## Inserting Encoded Passwords into Method Arguments

You can encode passwords to use the resulting strings as method 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. The **Password Encoder** enables you to encode your passwords.

**To encode a password:**

1 From the **Windows** menu, choose **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Password Encoder**. The Password Encoder dialog box opens.




2 Enter the password in the **Password** box.

3 Click **Generate**. The Password Encoder encrypts the password and displays it in the **Encoded String** box.

4 Use the **Copy** button to copy and paste the encoded value into the Data Table.

5 Repeat the process for each password you want to encode.

6 Click **Close** to close the Password Encoder.

## Defining an Output Value for Your Step

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, QuickTest retrieves each value selected for output and stores it in the specified location for use later in the run session.

When you create a new output value step, QuickTest assigns a default definition to each value selected for output. 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 tab 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. If you select a component parameter, the details for the output parameter are read-only. You can modify the parameter details in the Parameters tab of the Business Component Settings dialog box. For more information, see "Defining Parameters for Your Component" on page 609.

If, after you specify an output value, you choose not to save the output value, you can cancel it. For more information, see "Canceling Output to a Parameter" on page 531.

**To configure output to a parameter:**

1 Click in the **Output** cell to create or edit an output to a parameter. Click the **Output** button or press CTRL + F11. The Output Options dialog box opens.


2 In the **Output Types** box, select either **Component parameter** or **Local parameter**. The **Details** area displays the options available for the selected component type.

---

**Note:** The **Component parameter** type is available only if an output component parameter is defined for the component. If you select a component parameter, the information displayed is read-only. For more information, see “Defining Parameters for Your Component” on page 609.

---

3 Select the required parameter from the **Name** box. If no local parameter is defined, then **p_Local** is the default parameter name displayed.

- ➤ You can create a new local parameter, if needed. For more information, see "Working with Parameters" on page 533.
- ➤ If you select a local parameter, specify the details for it. For more information, see "Working with Parameters" on page 533.
- ➤ If you select a component parameter, its details are read-only.

4 Click **OK**. The **Output** cell displays the parameter to which the output value will be saved.

---

**Tip:** 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.

---

### Canceling Output to a Parameter

If you do not want to store the output value for a component step, you can cancel it.

**To cancel output to a parameter:**

Click in the **Output** cell. Then click the **Cancel** button  or press the DELETE key to cancel output to the parameter.

## Adding Other Types of Steps to Your Component

In addition to adding standard steps to your component in the Keyword View, you can also insert the following special types of steps using the relevant options from the **Insert** menu. Each step is entered as a row in the Keyword View, and you can then modify it as described in "Modifying the Parts of a Step" on page 532.

- You can insert a checkpoint step. For more information, see "Understanding Checkpoints" on page 553.
- You can insert an output value step. For more information, see "Outputting Values" on page 571.
- You can insert comments in steps to separate parts of a component to add details about a specific part. For more information, see "Adding Comments" on page 786.

## Modifying the Parts of a Step

You can modify any part of a step in the Keyword View. For example, you can change the test object on which the step is performed or change the operation to be performed in the step.

When working in the Keyword View, you can use the standard editing commands (**Cut**, **Copy**, **Paste** and **Delete**) in the **Edit** menu or in the context-sensitive menu to make it easier to modify your steps.

---

**Tip:** You can copy and paste, or drag and drop steps to move them to a different location in an action.

---

To modify a step, click in the cell containing the part of the step you want to modify and specify the content of the cell. Each cell in the step row represents a different part of the step. For more information, see "Adding a Step to Your Component" on page 514.

## Working with Parameters

You can define input parameters that pass values into your business 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. Local output parameters can be returned by an operation or component step for use within the same component. Local parameter output values can be viewed in the business process test results.

You define local parameters in the Business Component Keyword View using the Configure Value Options dialog box for input parameters and the Output Options dialog box for output parameters. 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 test results.

You define component parameters in the Parameters tab of the Business Component Settings dialog box or in the Quality Center Business Components module.

This section describes how to configure local parameters and parameterize input and output values using local and component parameters. For information on configuring component parameters, see "Defining Parameters for Your Component" on page 609.

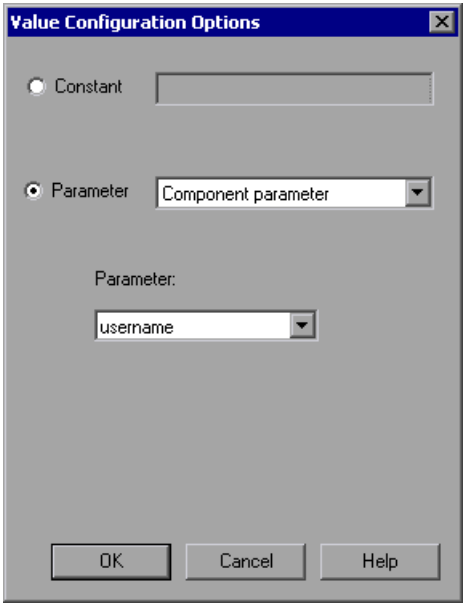
After you define a parameter you can use it to parameterize a value. Alternatively, you can apply a constant value to the parameter by typing it directly in the **Value** cell.

## Parameterizing Input Values

In the **Value** cell, you can parameterize input values for a step using a local parameter or a component parameter.

**To parameterize an input value using a local parameter:**

**1** In the **Value** cell, click the parameterization button or press CTRL + F11. The Value Configuration Options dialog box opens.

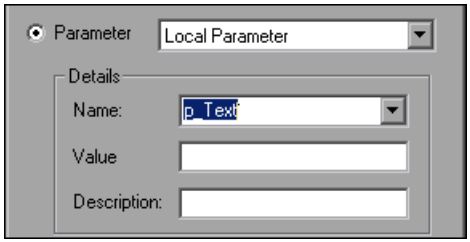



---

**Note:** 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 tab of the Business Component Settings dialog box.

---

**2** In the **Parameter** box, select **Local parameter**. The details for the local parameter type are displayed.




**3** 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. If you do not specify a value, QuickTest assigns a default value, as follows:

| Type of Value | QuickTest Default Value |
|---|---|
| String | Empty string |
| Boolean | True |
| Date | The current date |
| Number | 0 |
| Password | Empty string |

- **Description.** Enter a brief description for the parameter.

**4** Click **OK**. The local parameter is displayed in the **Value** cell of your step. When the component is run, it will use the value specified in the parameter for the step.

---

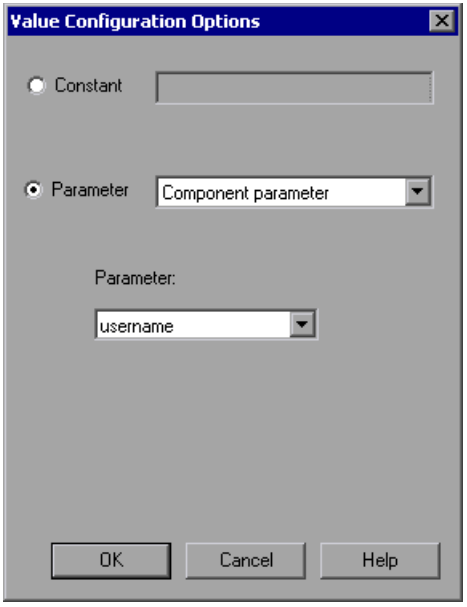
**Tips:**

- You can cancel the parameterization of a value by selecting the **Constant** option in the Value Configuration Options dialog box and entering a constant value.
- 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.

---

**To parameterize an input value using a component parameter:**

**1** In the **Value** cell, click the parameterization button  or press CTRL + F11. The Value Configuration Options dialog box opens.

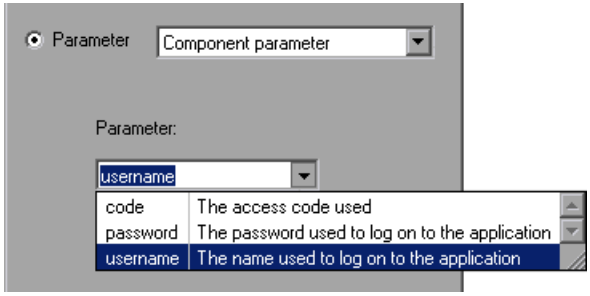


---

**Note:** 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 input parameter displayed in the Parameters tab of the Business Component Settings dialog box.
If no component parameter is defined, you must define one before you can use it to parameterize an input value. For more information, see "Defining Parameters for Your Component" on page 609.

---

**2** In the **Parameter** box, 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.




**3** Click **OK**. The component parameter is displayed in the **Value** cell of your step. When the component is run, it will use the value specified in the parameter for the step.

**Tips:**

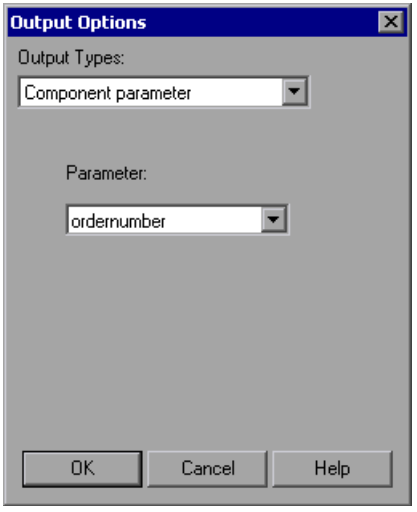
- You can cancel the parameterization of a value by selecting the **Constant** option in the Value Configuration Options dialog box and entering a constant value.
- If you click in the **Value** cell after you define a component parameter for it, the  icon is displayed in each part of the cell for which a component parameter is defined.

## Parameterizing Output Values

You can parameterize output values for a step using a local parameter or a component parameter, in the step **Output** cell. You can then use the output parameter value as an input value in a later step in the component, or in a later component in the business process test.

**To parameterize an output value using a local parameter:**

**1** In the **Output** cell, click the output value button  or press CTRL + F11. The Output Options dialog box opens.

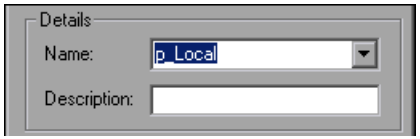


---

**Note:** 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 tab of the Business Component Settings dialog box.

---

**2** In the **Output Types** box, select **Local parameter**. The details for the local parameter type are displayed.




**3** Specify the property details for the local parameter:

- **Name.** Enter a meaningful name for the parameter or choose one from the list.
- **Description.** Enter a brief description for the parameter.

**4** Click **OK**. The local parameter is displayed in the **Output** cell of your step. When the component is run, it will output the value to the output parameter specified for the step.

---

**Tip:** If you click in the **Output** 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.

---

**To parameterize an output value using a component parameter:**

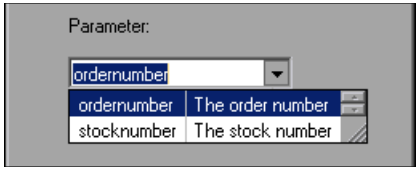
**1** In the **Output** cell, click the output value button or press CTRL + F11. The Output Options dialog box opens.


---

**Note:** 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 tab of the Business Component Settings dialog box.
If no component parameter is defined, you must define one before you can use it to parameterize an output value. For more information, see "Defining Parameters for Your Component" on page 609.

---

**2** In the **Parameter** box, select the component parameter in which to store the output 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.




**3** Click **OK**. The component parameter is displayed in the **Output** cell of your step. When the component is run, it will output the value to the output parameter specified for the step.

---

**Tip:** 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.

---

## Working with Comments

A **Comment** is a free text entry that can be entered in a business component. The  icon indicates a comment in the Keyword View. You can use comments for several purposes. For example, you may want to plan steps to be included in a business component before your application is ready to be tested. Then, when your application is ready to be tested, you can use your plan to verify that every item that needs to be tested is included in the component steps.

You may want to add comments to a component to improve readability and make it easier to update. For example, you may want to add a comment before each section of a component to specify what that section includes.

After you add a comment, it is always visible in your component, as long as one or more columns are displayed. For information on selecting columns to display, see "Defining Keyword View Display Options" on page 545. In addition, as you scroll from side to side across the grid, the comment can always be seen. QuickTest does not process comments when it runs a component.

---

**Note:** After you insert a comment, you cannot change it to a step.

---

**To add a comment to your component:**

1. Choose **Insert** > **Comment**, click in the **Item** cell and choose **Comment** from the displayed list, or right-click on a component step and select **Insert Comment**. A comment row is added below the selected step.
2. Enter text in the Comment row. If you do not enter text, QuickTest deletes the comment when the cursor focus is removed.

**To modify an existing comment:**

Double-click the comment in the **Comment** column. The text box becomes a free text field. Alternatively, you can click the  icon, which acts as a toggle, making the comment either editable or read-only.

**To delete a comment:**

1. Select the comment and choose **Edit** > **Delete**, press the DELETE key on your keyboard, or right-click and select **Delete** from the context-sensitive menu.
2. Click **Delete Comment** to confirm. The comment is permanently removed from the business component.

# Managing Component Steps

You can move a component step before or after any other step or comment in a component. You can also delete it if it is no longer required.

## Moving a Component Step

You can move a step to a different location within a component, as needed.

**To move a step in the component:**

- In the **Item** column, drag the step up or down and drop it at the required location. When you drag a selected step, a line is displayed, enabling you to see the location to which the step will be moved.
- Copy or cut the step to the Clipboard and then paste it in the required location. You can use **Edit** > **Copy** or CTRL + C to copy the step, **Edit** > **Cut** or CTRL + X to cut the step, and **Edit** > **Paste** or CTRL + V to paste the step.

## Deleting a Component Step

You can delete a component step, if required. Before you delete a step, make sure that removing it will not prevent the component from running correctly.

---

**Note:** You cannot delete a step if one of its cells is in edit mode.

---

**To delete a step:**

1. Select the step that you want to delete and choose **Edit** > **Delete**, press the DELETE key, or right-click on the step and select **Delete** from the context-sensitive menu. A warning message displays.
2. Click **Delete Step** to confirm. The step is deleted from the component.

## Using Keyboard Commands in the Keyword View

If you prefer to use your keyboard, you can use the following keyboard commands to navigate within the Keyword View:

- Press F8 to add a new step below the currently selected step.
- Press SHIFT+F8 to add a new step after a conditional or loop block.
- Press F7 to use the Step Generator to add a new step below the selected step.
- The TAB and SHIFT+TAB keys move the focus left or right within a single row, unless you are in a cell that is in edit mode. If so, press ENTER to exit edit mode, and then you can use the TAB keys.
- When a cell containing a list is selected:
  - You can press SHIFT+F4 to open the list for that cell.
  - You can change the selected item by using the up and down arrow keys. In the **Item** column, the list must be open before you can use the arrow keys.
  - You can type a letter or sequence of letters to move to a value that starts with the typed letters. The typed sequence is highlighted in white.

# Defining Keyword View Display Options

You can choose how you want to display the information in the Keyword View using the Keyword View Options dialog box. You can customize the display of the Keyword View columns, fonts, and colors. The options you set remain in effect for all tests in all subsequent sessions on your computer.

## Displaying Keyword View Columns

You can use the Columns tab of the Keyword View Options dialog box to specify which columns you want to display in the Keyword View. You can also specify the order in which the columns are displayed.

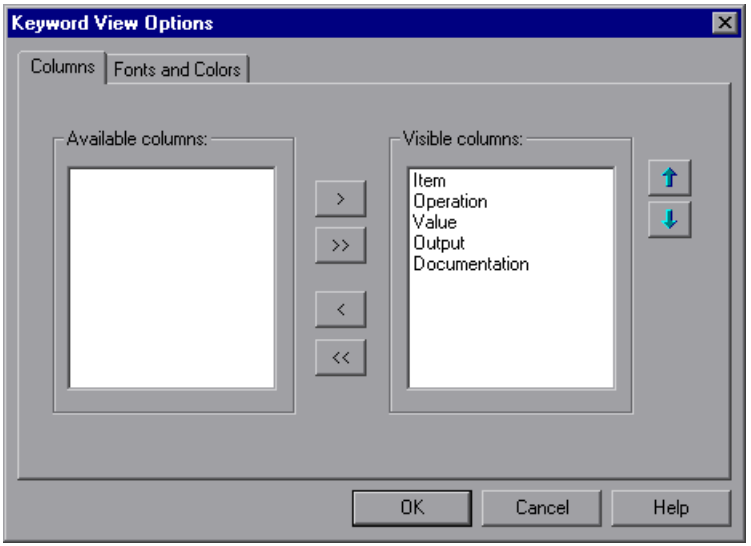
---

**Tip:** You can display only the **Documentation** column by right-clicking the column header row and choosing **Documentation Only** from the displayed menu. You can then print the Keyword View for use as instructions for manual testing. For more information on printing from the Keyword View, see "Printing a Test" on page 334.

---

**To specify the Keyword View columns to display:**

**1** Choose **Tools** > **View Options**. The Keyword View Options dialog box opens.




The **Available columns** list shows columns not currently displayed in the Keyword View. The **Visible columns** list shows columns currently displayed in the Keyword View.

**2** Double-click column names or choose column names and click the arrow buttons (> and <) to move them between the **Available columns** and **Visible columns** lists.

---

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

---

**3** In the **Visible columns** list, set the order in which columns appear in the Keyword View by selecting one or more columns and then using the up and down arrow buttons.

---

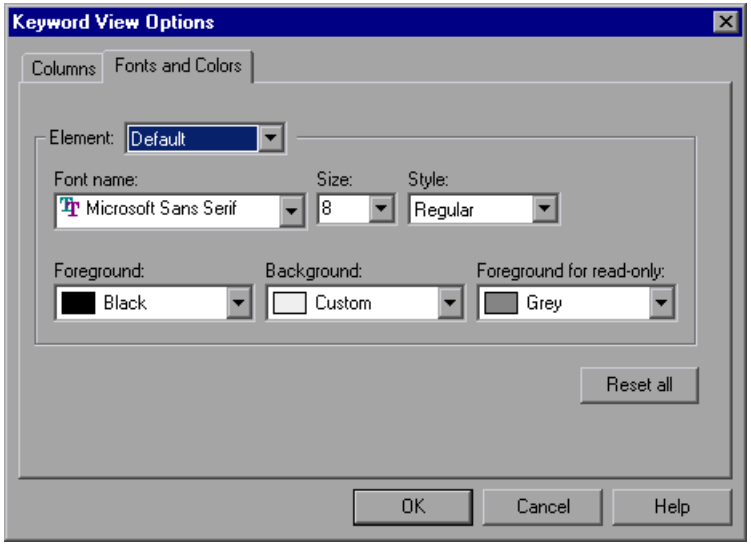
**Note:** The order of the columns in the Keyword View does not affect the order in which the cells need to be completed for each step. For example, if you choose to display the **Operation** column to the left of the **Item** column, you still need to select the item first, and only then is the **Operation** column list refreshed to match the selection you made in the **Item** column.

---

**4** Click **OK** to close the dialog box and apply the new column display.

## Setting Keyword View Fonts and Colors

You can use the Fonts and Colors tab of the Keyword View Options dialog box to specify different text and color display options for different elements in the Keyword View.



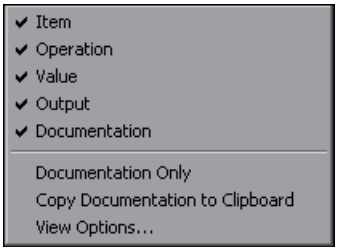
The Fonts and Colors tab includes the following options:

| Option | Description |
|---|---|
| **Element** | 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:<br>➤ **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.<br>➤ **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).<br>➤ **Default**. All rows and text in the Keyword View (except for the elements listed below).<br>➤ **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.<br>**Note:** When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your component may not be correctly displayed in the Keyword View. However, the 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. |

| Option | Description |
|---|---|
| **Style** | Enables you to modify the font style used for text in the selected element. You can select **Regular**, **Bold**, **Italic**, or **Underline** font styles. You cannot change the font style for **Alternate Rows** or **Selected Row** elements. |
| **Foreground** | Enables you to modify the text color for the selected element. You cannot change the foreground color for **Alternate Rows**. |
| **Background** | Enables you to modify the row color for the selected element. |
| **Foreground for read-only** | Enables you to modify the text color for rows that are read-only. This option cannot be changed for **Alternate Rows**. |
| **Reset all** | Resets all Fonts and Colors tab options to the default settings. |

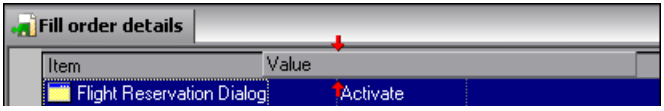
## Tips for Working with the Keyword View

- You can display or hide specific columns by right-clicking the column header row in the Keyword View and then selecting or deselecting the required column name from the displayed menu.




  You can display only the **Documentation** column, for example, if you want to print the steps for use as instructions for manual testing, by selecting **Documentation Only**. The **Documentation** column and any comments defined in the component are displayed.

- You can rearrange columns by dragging a column header to its new location in the Keyword View. Red arrows are displayed when the column header is dragged to an available location.



# Working with Breakpoints in the Keyword View

You can insert and remove breakpoints in the Keyword View.

**To insert a breakpoint in the Keyword View:**

- Click in the left margin at the point where you want to insert the breakpoint.
- Select a step and press F9.
- Choose **Debug** > **Insert/Remove Breakpoint**.

  A red breakpoint icon is displayed.

**To remove a breakpoint from the Keyword View:**

- Click the breakpoint icon.
- Select a step and press F9.
- Choose **Debug** > **Insert/Remove Breakpoint**.

For more information on breakpoints, see “Using Breakpoints” on page 690.

# 16

# Understanding Checkpoints

You can check objects in your application to ensure that they function properly.

**This chapter includes:**



## About Understanding Checkpoints

QuickTest enables you to add checks to your test or component. A **checkpoint** is a verification point that compares a current value for a specified property with the expected value for that property. This enables you to identify whether your application is functioning correctly. You can perform standard checkpoints and bitmap checkpoints on component steps.

When you add a checkpoint, QuickTest adds a checkpoint to the current row in the Keyword View. By default, the checkpoint name receives 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 component, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails. You can view the results of the checkpoint in the Test Results window.

You insert a checkpoint step by choosing the relevant checkpoint option from the **Insert** menu while creating or modifying your component. (You can insert bitmap checkpoints only during a recording session.) When you create a checkpoint in a component, the displayed Checkpoint Properties dialog box opens in **Simple Mode**, which shows only the properties and values that can be viewed in Quality Center. For more information on this dialog box, see "Understanding the Checkpoint Properties Dialog Box" on page 559.

You can view or edit advanced checkpoint properties by clicking the **Advanced Mode** button. However, if the checkpoint contains advanced properties, and a Quality Center user views its properties in Quality Center, a disclaimer opens indicating that some properties are checked but not shown. The Quality Center user does not have the option to view or edit the advanced properties.

## Adding New Checkpoints to a Component

You can add checkpoints while creating or editing your component. It is generally more convenient to define checkpoints after creating the initial component.

**To add new checkpoints while recording your component:**

Use the commands in the **Insert** > **Checkpoint** menu, or click the **Insert Checkpoint** button on the toolbar. This displays a menu of checkpoint options that are relevant to the selected step.

## Understanding Types of Checkpoints

You can insert the following checkpoint types to check various objects in an application.

- **Standard Checkpoint** checks the property value of an object in your application. The standard checkpoint checks a variety of objects such as buttons, radio buttons, combo boxes, lists, and so forth. For example, you can check that a radio button is activated after it is selected or you can check the value of an edit box.

  Standard checkpoints are supported for all add-in environment.

  For more information on standard checkpoints, see Chapter 17, "Checking Object Property Values."

- **Bitmap Checkpoint** checks an area of your application as a bitmap. For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. You can record the new map that is displayed after one click on the control key that zooms in the map. Using the bitmap checkpoint, you can check that the map zooms in correctly.

  You can create a bitmap checkpoint for any area in your application, including buttons, text boxes, and tables.

  Bitmap checkpoints are supported for all add-in environments.

  For more information on bitmap checkpoints, see Chapter 18, "Checking Bitmaps."

# 17

# Checking Object Property Values

By adding standard checkpoints to your components, you can compare object property values in your application with the expected values.

**This chapter includes:**



## About Checking Object Property Values

You can check the object property values in your application using standard checkpoints. Standard checkpoints compare the expected values of object properties to the object's current values during a run session. You can create standard checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

---

**Note:** Users in Quality Center cannot create, edit, or rename checkpoints in components.

---

# Creating Standard Checkpoints

You can check that a specified object in your application has the property values you expect, by adding a standard checkpoint step to your component while recording or editing the component. To set the options for a standard checkpoint, you use the Checkpoint Properties dialog box.

---

**Note:** You cannot create image, table, or (Web) page checkpoints in a keyword component. These special checkpoint types are only available for tests. However, if you select a Web page or any table object when creating a standard checkpoint for your component, you will be able to check their object properties just like any other object.

---

**To add a standard checkpoint while recording:**

**1** While in a recording session, choose **Insert** > **Checkpoint** > **Standard Checkpoint**, or click the **Insert Checkpoint or Output Value** toolbar button.

The QuickTest window is hidden, and the pointer changes into a pointing hand.

---

**Note:** You can hold the left CTRL key to change the pointing hand into a standard pointer, and then change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

---

**2** Click the object you want to check. The Object Selection - Checkpoint Properties dialog box opens. .

**3** Select the item you want to check from the displayed object tree. The tree item name corresponds to the object's class.

**4** Click **OK**. The Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box" on page 559.

6 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View.

**To add a standard checkpoint while editing:**

1 Select the step where you want to add the checkpoint and choose **Insert** > **Checkpoint** > **Standard Checkpoint**.

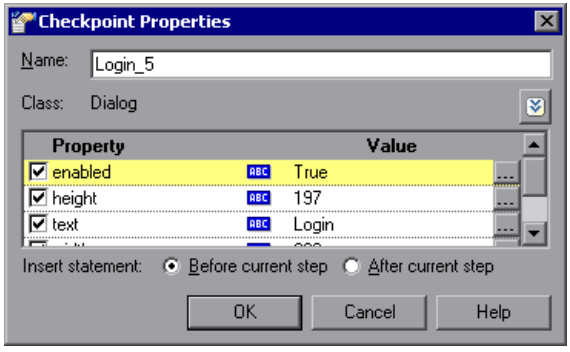
The Checkpoint Properties dialog box opens.

2 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box" on page 559.

3 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View.

## Understanding the Checkpoint Properties Dialog Box

In the Checkpoint Properties dialog box, you can specify which properties of the object to check, and edit the values of these properties. The Checkpoint Properties dialog box for keyword components has two modes, **Simple Mode** and **Advanced Mode**. The mode that opens is dependent on whether any advanced properties are selected in the checkpoint.



For more information, see:

- "Understanding the Checkpoint Properties Dialog Box - Simple Mode" on page 560
- "Editing the Expected Value of an Object Property" on page 563
- "Understanding the Checkpoint Properties Dialog Box - Advanced Mode" on page 563

## Understanding the Checkpoint Properties Dialog Box - Simple Mode

For new checkpoints, the Checkpoint Properties dialog box always opens in Simple Mode. If you open the dialog box for an existing checkpoint and no advanced checkpoint properties are selected, the Checkpoint Properties dialog box also opens in Simple Mode. In Simple Mode, only the basic properties and expected values of the object are shown. To view or edit advanced checkpoint properties, click the **Advanced Mode** button.

---

**Note:** The advanced properties of checkpoints cannot be viewed or edited in Quality Center. Therefore, if one or more advanced properties are selected in a checkpoint, and a Quality Center user views its properties in Quality Center, the dialog box displays text indicating that some properties selected for checking are not shown.

---

## Identifying the Checkpoint

The top part of the dialog box displays information on the checkpoint:

| Information | Description |
|---|---|
| **Name** | The name of the checkpoint. By default, the checkpoint name is the same as the name of the test object on which the checkpoint was created. You can specify a different name for the checkpoint or accept the default name.<br>If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:<br>" := @@ |
| **Class** | The type of object. The Class of the object is not displayed in Quality Center for keyword components. |

## Selecting the Object Property to Check

The properties for the object are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

| Pane Element | Description |
|---|---|
| **Check box** | For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly.<br>To check a property, select the corresponding check box.<br>To exclude a property check, clear the corresponding check box. |
| **Type** | The  icon indicates that the value of the property is currently a constant.<br>The  icon indicates that the value of the property is currently a parameter. |

| Pane Element | Description |
|---|---|
| **Property** | The name of the property. |
| **Value** | The expected value of the property. For more information on modifying the value of a property, see "Editing the Expected Value of an Object Property" on page 563. |

### Inserting the Checkpoint in Your Component

The **Insert statement** option specifies when to perform the checkpoint in the component.

- Choose **Before current step** if you want to check the value of the object property before the highlighted step is performed.
- Choose **After current step** if you want to check the value of the property after the highlighted step is performed.

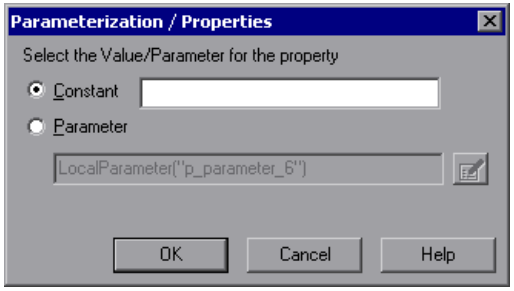
---

**Note:** The **Insert statement** option is not available when adding a checkpoint during recording or when modifying an existing object checkpoint. It is available only when adding a new checkpoint to an existing component while editing it.

---

## Editing the Expected Value of an Object Property

When you click the Browse button for a property [...] in the Checkpoint Properties dialog box, the Parameterization / Properties dialog box opens, in which you can set the property value as a **Constant** or a **Parameter**. The default is **Constant**.




- **Constant.** A value that is defined directly in the step and remains unchanged when the component runs.

  If you select **Constant**, you can edit the value directly in the **Constant** box.

- **Parameter.** A value that is defined or generated separately from the step and is retrieved when the specific step runs.

If you select **Parameter** for a value that is already parameterized, the **Parameter** box displays the current parameter definition for the value. If you select **Parameter** for a value that is not yet parameterized, you can click the **Parameter Options** button to open the Parameter Options dialog box.

Specify the property details for the parameter by selecting a different parameter type or modifying the parameter value settings. For more information on using parameters in your components, see "Working with Parameters" on page 533.

## Understanding the Checkpoint Properties Dialog Box - Advanced Mode

If advanced checkpoint properties are selected, the Checkpoint Properties dialog box opens in Advanced Mode, in which all the supported properties and expected values of the object are shown. To view or edit simple checkpoint properties, click the **Simple Mode** button.

The bottom part of the Checkpoint Properties dialog box in Advanced Mode contains the following additional options:

➤ **Checkpoint timeout.** Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

➤ **Find in Repository.** To view the checkpoint in its repository, click the **Find in Repository** button. (This option is not available when creating a new checkpoint. It is available only when editing an existing checkpoint.)

For more information, see "Understanding the Object Repository Window" on page 120.

## Modifying Checkpoints

You can modify the settings of existing checkpoints.

**To modify a checkpoint:**

1 Select the step containing the checkpoint and choose **Edit** > **Step Properties** > **Checkpoint Properties**. The relevant checkpoint dialog box opens.

2 Modify the properties and click **OK**. For more information, see "Understanding the Checkpoint Properties Dialog Box" on page 559.

# 18

# Checking Bitmaps

QuickTest enables you to compare objects in an application by matching captured bitmaps.

**This chapter includes:**



## About Checking Bitmaps

You can check an area of an application as a bitmap. QuickTest captures the **visible** part of the specified object as a bitmap (QuickTest does not capture any part that is scrolled off of the screen, for example), and inserts a checkpoint in the component.

When you run the component, QuickTest compares the object in the application with the bitmap stored in the checkpoint. If there are differences, QuickTest captures a bitmap of the actual object and displays it with the expected bitmap in the details portion of the Test Results window. By comparing the two bitmaps (expected and actual), you can identify the nature of the discrepancy. For more information on test results of a checkpoint, see "Viewing Checkpoint Results" on page 670.

For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. You can record the new map that is displayed after one click on the control key that zooms in the map. Using the bitmap checkpoint, you can check that the map zooms in correctly.

You can create bitmap checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

---

**Note:** The results of bitmap checkpoints may be affected by factors such as operating system, screen resolution, and color settings.

---

## Checking a Bitmap

You insert bitmap checkpoints while recording a component.

**To create a bitmap checkpoint while recording:**

**1** Choose **Insert** > **Checkpoint** > **Bitmap Checkpoint**, or click the **Insert Checkpoint or Output Value** button and choose **Bitmap Checkpoint**.

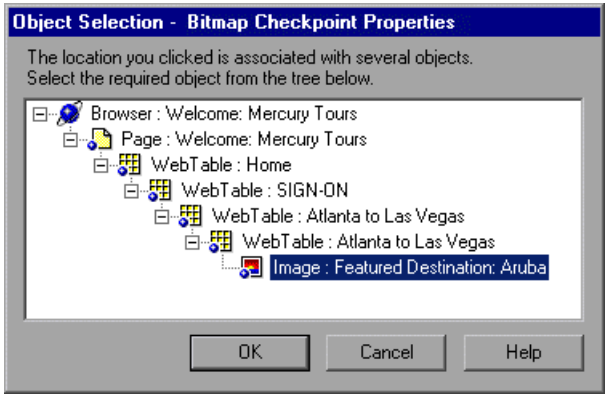
The QuickTest window is hidden, and the pointer turns into a pointing hand.

---

**Tip:** You can hold the left CTRL key to change the pointing hand into a standard pointer, and then change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

---

**2** Click an object to check in your application. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



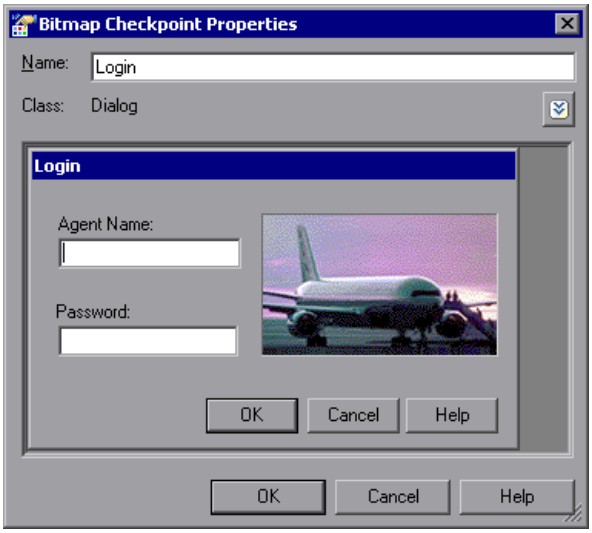

**3** Select an object from the tree on which to create a bitmap checkpoint.

---

**Tip:** If you want to create a bitmap checkpoint that contains multiple objects, you should select the highest level object that includes all the objects to include in the bitmap checkpoint.

---

**4** Click **OK**. The Bitmap Checkpoint Properties dialog box opens in Simple Mode.




A bitmap of the object you selected in the previous step is displayed in the dialog box.

**Advanced Mode** enables you to select a specific sub-area of the bitmap, set RGB and pixel tolerances, set a checkpoint timeout, and view the checkpoint in its repository. To view or edit bitmap properties, click the **Advanced Mode** button. For more information, see "Understanding the Bitmap Checkpoint Properties Dialog Box - Advanced Mode" on page 569.

---

**Note:** Quality Center users do not have the option to view or edit bitmap properties. Therefore, if one or more bitmap properties are selected in a checkpoint, and a Quality Center user views its properties in Quality Center, the dialog box displays text indicating that some properties selected for checking are not shown.

---

**5** In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the same as the name of the test object on which the checkpoint was created.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:
" := @@

---

**Note:** The **Class** area displays the type of test object on which the checkpoint was created.

---

**6** Click **OK** to add the bitmap checkpoint to your component. A checkpoint statement is added for the selected object in the Keyword View.

## Understanding the Bitmap Checkpoint Properties Dialog Box - Advanced Mode

Advanced Mode in the Bitmap Checkpoint Properties dialog box includes the following buttons and options:

- **Select Area.** Enables you to check a specific area of the object. Click the button and use the crosshairs pointer to specify the area you want to select. This instructs QuickTest to check only the selected area and to ignore the remainder of the bitmap.
- **Save only selected area.** Enables you to save only the selected area of the object (to save disk space).
- **RGB tolerance.** Enables you to define a tolerance for the RGB (Red, Green, Blue) properties of a bitmap checkpoint. Defining a tolerance determines the percent by which the RGB values of the actual object are allowed to be different from those of the expected object and allow the checkpoint to pass.

- **Pixel tolerance.** Enables you to define a pixel tolerance that determines the amount by which the pixels of the actual object can differ from those of the expected object and allow the checkpoint to pass. You can choose to define the pixel tolerance as a specific number of pixels or as a percentage of the total pixels in the object.
- **Checkpoint Timeout.** Enables you to define the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

- **Find in Repository.** To view the checkpoint in its repository, click the Find in Repository button. (This option is not available when creating a new checkpoint. It is available only when editing an existing checkpoint.)

# 19

# Outputting Values

QuickTest enables you to retrieve values in your component and store them in output value objects. You can subsequently retrieve these values and use them as input at a different stage in the run session.

**This chapter includes:**



## About Outputting Values

An **output value** step is a step in which one or more values are captured at a specific point in your component and stored for the duration of the run session. The values can later be used as input at a different point in the run session.

When you create output value steps, you can determine where the values are stored during the run session and how they can be used. During the run session, QuickTest retrieves each value at the specified point and stores it in the specified location. When the value is needed later in the run session, QuickTest retrieves it from this location and uses it as required.

Output values are stored only for the duration of the run session. When the run session is repeated, the output values are reset.

---

**Note:** After the run session, you can view the output values retrieved during the session as part of the session results. For more information, see "Viewing Parameterized Values and Output Value Results in the Test Results Window" on page 674.

---

When you create an output value in a keyword component, the Output Value Properties dialog box opens in Simple Mode, which shows only the properties and values that can be viewed in Quality Center. For more information on this dialog box, see "Defining Standard Output Values" on page 575.

You can view or select advanced output properties by clicking the **Advanced Mode** button. However, the user in Quality Center does not have the option to view or edit the advanced properties. Therefore, if the output value object is set to retrieve advanced properties, and a Quality Center user views its setting in Quality Center, the Output Value Properties dialog box displays text indicating that some properties are selected for output but not shown.

## Creating 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.

For more information on standard output values, see "Outputting Property Values" on page 573.

### Viewing and Editing Output Values

When you insert an output value step in your test, the Keyword View shows the step with Output displayed in the **Operation** column and CheckPoint displayed in the **Value** column, followed by the name assigned to the output value.

## Outputting Property Values

You can use standard output values to output the property values of most objects.

You can create standard output values while recording or editing your component.

---

**Note:** You cannot create image, table or (Web) page output values in a component. These special output value types are only available for tests. However, if you select a Web page or any table object when creating a standard output value for your component, you will be able to check their object properties just like any other object.

---

**To create standard output values while recording:**

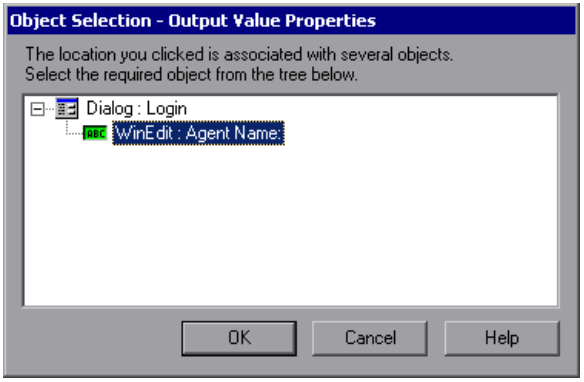
**1** Choose **Insert** > **Output Value** > **Standard Output Value**. Alternatively, you can click the arrow beside the **Insert Checkpoint or Output Value** button on the toolbar and select **Standard Output Value**. The pointer changes into a pointing hand.

---

**Tip:** You can hold the left CTRL key to change the pointing hand into a standard pointer, and then change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

---

**2** In your application, click the object for which you want to specify an output value. If the location you clicked is associated with more than one object, the Object Selection – Output Value Properties dialog box opens.



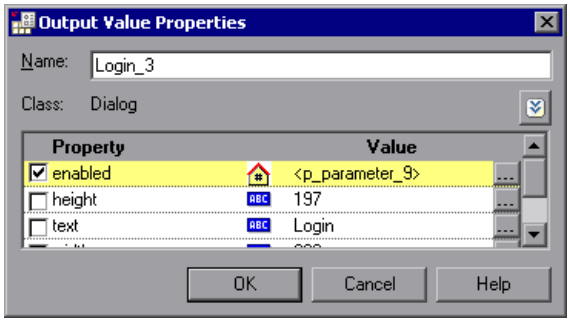

**3** In the Object Selection dialog box, select the object for which you want to specify an output value, and click **OK**. The Output Value Properties dialog box opens for the selected object.

**4** Specify the property values to output and their settings. For more information, see “Defining Standard Output Values” on page 575.

**5** When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your component.

## Defining Standard Output Values

The Output Value Properties dialog box enables you to choose which property values to output and to define the settings for each value that you select. The Output Value Properties dialog box has two modes, Simple Mode and Advanced Mode. The mode that opens is dependent on whether any advanced properties are selected for output.




When inserting a new output value step, the dialog box always opens in Simple mode. If you open the dialog box for an existing output value object and no advanced checkpoint properties are selected, the Output Value Properties dialog box opens in Simple Mode. In Simple mode, only the basic properties are shown. To view or select advanced properties, click the **Advanced Mode** button.

---

**Note:** The user in Quality Center does not have the option to view or edit the advanced properties. Therefore, if the output value has advanced properties selected, and a user views its properties in Quality Center, the Output Value Properties dialog box displays text that indicates that some properties are selected but not shown.

---

You can select a number of properties to output for the same object and define the output settings for each property value before closing the dialog box. When the output value step is reached during the run session, QuickTest retrieves all of the specified property values.

### Identifying the Output Value

The top part of the dialog box displays information on the output value:

| Item | Description |
|---|---|
| **Name** | The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name.<br>If you rename the output value, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:<br>" := @@ |
| **Class** | The type of test object. |

### Selecting the Property Values to Output

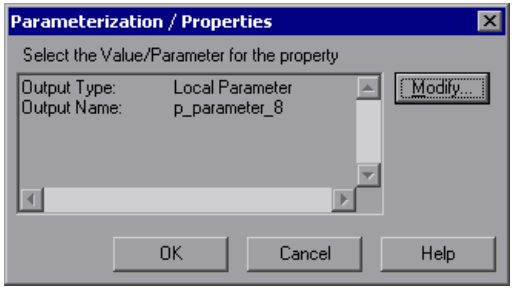
The upper part of the dialog box contains a pane that lists the properties of the selected object, with their values and types. This pane contains the following items:

| Pane Element | Description |
|---|---|
| **Check box** | To specify a property to output, select the corresponding check box. You can select more than one property for the object and specify the output options for each property value you select. |
| **Type** | The  icon indicates that the value of the property is currently a constant.<br>The  icon indicates that the value of the property is currently a parameter. |

| Pane Element | Description |
|---|---|
| **Property** | The name of the property. |
| **Value** | The current value of the property. For more information, see "Specifying the Output Settings for a Property Value" on page 577. |

### Specifying the Output Settings for a Property Value

When you click the Browse button for a selected property ... in the Output Value Properties dialog box, the Parameterization / Properties dialog box opens, in which the output definition for the selected property value is displayed.




When you select a property value to output, you can:

- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see "Specifying the Output Type and Settings" on page 578.
- accept the displayed output definition by clicking **OK**.

### Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see "Selecting the Location for the Output Value Step" on page 578.

### Find in Repository

To view the output value in its repository, click the Find in Repository button. (This option is not available when creating a new output value. It is available only when editing an existing output value.)

## Specifying the Output Type and Settings

The output type and settings that you define for each value determine where it is stored and how it can be used during the run session. When the output value step is reached, QuickTest retrieves each value selected for output and stores it in the specified location for use later in the run session. For more information, see "Selecting the Location for the Output Value Step" on page 578.

### Selecting the Location for the Output Value Step

When you create output values while editing a component, the **Insert statement** area is displayed at the bottom of the dialog box.

By default, QuickTest inserts the new output value step before the current step (the step you selected when you chose the **Output Value** option). You can instruct QuickTest to insert the new output value step after the current step, by selecting the **After current step** option.

---

**Note:** This option is not available while recording. QuickTest automatically inserts the new output value step after the previously recorded step. It is also not available when modifying an existing output value step.

---

# Part V

## Configuring Settings

# 20

# Setting Global Testing Options

You can control how QuickTest works with components by setting global testing options.

**This chapter includes:**



## About Setting Global Testing Options

Global testing options affect how you record and run components, as well as the general appearance of QuickTest. For example, you can choose not to display the Start Page when QuickTest starts, or you can set the timing-related settings used by QuickTest when running a component. The values you set remain in effect for all components and for subsequent testing sessions. You can set global testing options using the Options dialog box (described on page 582) or by inserting statements in the Expert View.

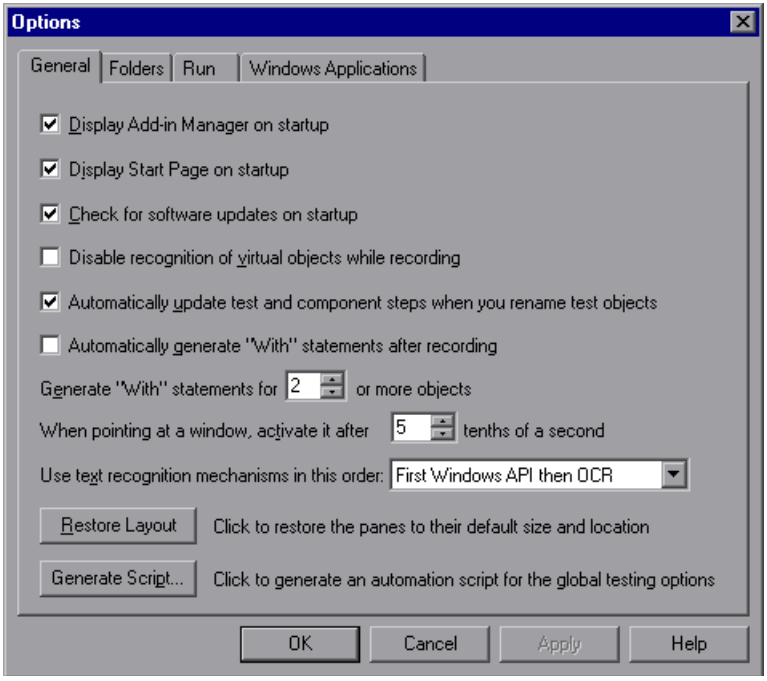
You can also set testing options that affect only the component currently open in QuickTest. For more information, see Chapter 21, "Working with Business Component Settings."

# Using the Options Dialog Box

You can use the Options dialog box to modify your global testing options. The values you set remain in effect for all subsequent QuickTest sessions.

**To set global testing options:**

**1** Choose **Tools** > **Options** or click the **Options** toolbar button. The Options dialog box opens. It is divided by subject into several tabbed pages.




**2** Select the required tab and set the options as necessary. For information on the available options in each tab, see the table below.

**3** Click **Apply** to apply your changes and keep the dialog box open, or click **OK** to save your changes and close the dialog box.

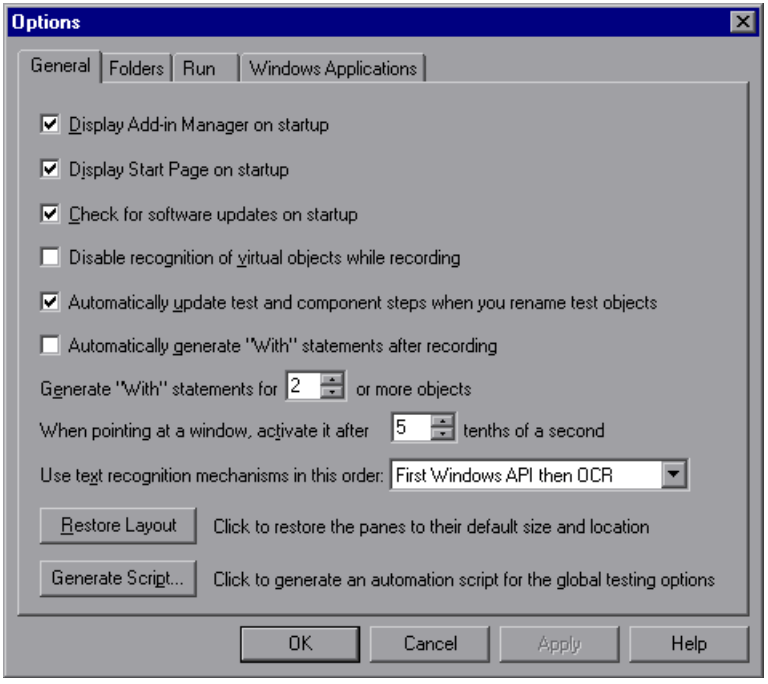
The Options dialog box contains the following tabbed pages:

| Tab Heading | Contains |
|---|---|
| **General** | Options for general component settings. For more information, see "Setting General Testing Options" on page 584. |
| **Folders** | Options for entering the folders (search paths) in which QuickTest searches for components or files that are specified as relative paths in dialog boxes and statements. For components, all files must be stored in the Quality Center subject path. For more information, see "Setting Folder Testing Options" on page 588. |
| **Run** | Options for running components. For more information, see "Setting Run Testing Options" on page 591. |
| **Windows Applications** | Options for configuring how QuickTest records and runs components for Windows applications. For more information, see the section on testing Windows-based applications in the *HP QuickTest Professional Add-ins Guide*. |

The Options dialog box may contain additional tabs, depending on the add-ins that are currently loaded. For more information, see the relevant section in the *HP QuickTest Professional Add-ins Guide*.

# Setting General Testing Options

The General tab options affect the general appearance of QuickTest and other general testing options.



The General tab includes the following options:

| Option | Description |
|---|---|
| **Display Add-in Manager on startup** | Determines whether the Add-in Manager is displayed when starting QuickTest. For information on working with the Add-in Manager, see the section on loading QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*. |
| **Display Start Page on startup** | Determines whether the Start Page is displayed when starting QuickTest. |
| **Check for software updates on startup** | Instructs QuickTest to automatically check for software updates each time it starts up. For more information, see the *HP QuickTest Professional User's Guide*. |
| **Disable recognition of virtual objects while recording** | Determines whether the defined virtual objects stored in the Virtual Object Manager are recognized while recording. This option is relevant only for tests. |
| **Automatically update test and component steps when you rename test objects** | Determines whether to automatically update test and component steps when you rename test objects in the local or shared object repository. For more information, see "Renaming Test Objects" on page 140. |
| **When pointing at a window, activate it after __ tenths of a second** | Specifies the time (in tenths of a second) that QuickTest waits before it sets the focus on an application window when using the pointing hand to point to an object in the application (for Object Spy, Recovery Scenario Wizard, and so forth). **Default = 5** |

| Option | Description |
|---|---|
| **Use text recognition mechanisms in this order** | Specifies the text recognition mechanism that QuickTest uses when capturing text for a text/text area checkpoint or output value step.<br>**Possible values:**<br>**First Windows API then OCR.** (Default) Instructs QuickTest to first try to retrieve text directly from the object using the Windows API-based mechanism. If no text can be retrieved (for example, because the text is part of a picture), QuickTest tries to retrieve text using the OCR (optical character recognition) mechanism. (Highly recommended when working with CJK languages.)<br>**First OCR then Windows API.** Instructs QuickTest to first try to retrieve text from the object using the OCR mechanism. If no text can be retrieved, then QuickTest uses its Windows API-based mechanism to retrieve text from the object.<br>**Use Only Windows API.** Instructs QuickTest to use only the Windows API-based mechanism (and not the OCR mechanism) to retrieve text from the object.<br>**Use Only OCR.** Instructs QuickTest to use only the OCR mechanism (and not the Windows API-based mechanism) to retrieve text from the object. (Required when working with Windows Vista.)<br>For more information on text recognition support in Windows-based environments, see the *HP QuickTest Professional Readme*. |

| Option | Description |
|---|---|
| **Restore Layout** | Restores the layout of the QuickTest window so that it displays the panes and toolbars in their default sizes and positions.<br>**Note:** QuickTest recalls your most recent window layout for each of its operating modes: view/edit, record, and run. For more information, see "Customizing the QuickTest Window Layout" on page 587. |
| **Generate Script** | Generates an automation script containing the current global testing options. For more information, see "Automating QuickTest Operations" on page 817, or see the *QuickTest Automation Reference* (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Automation**). |

## Customizing the QuickTest Window Layout

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

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

**To set the QuickTest layout for each mode:**

1. Open a new or existing 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.
6. Enter a breakpoint before the first step in the test. This enables you to arrange the layout during the run session. For information on how to set a breakpoint, see "Setting Breakpoints" on page 691.
7. Run your component.
8. When QuickTest reaches the breakpoint, set all of your layout preferences for the run mode.
9. Stop the run session.
10. Set all of your layout preferences for the view/edit mode.

The layouts for all of these modes are now set. QuickTest applies the relevant layout each time it enters one of these modes.

## Setting Folder Testing Options

The Folders tab enables you to enter the folders (search paths) in which QuickTest searches for components or files. All files must be stored in the Quality Center subject path.

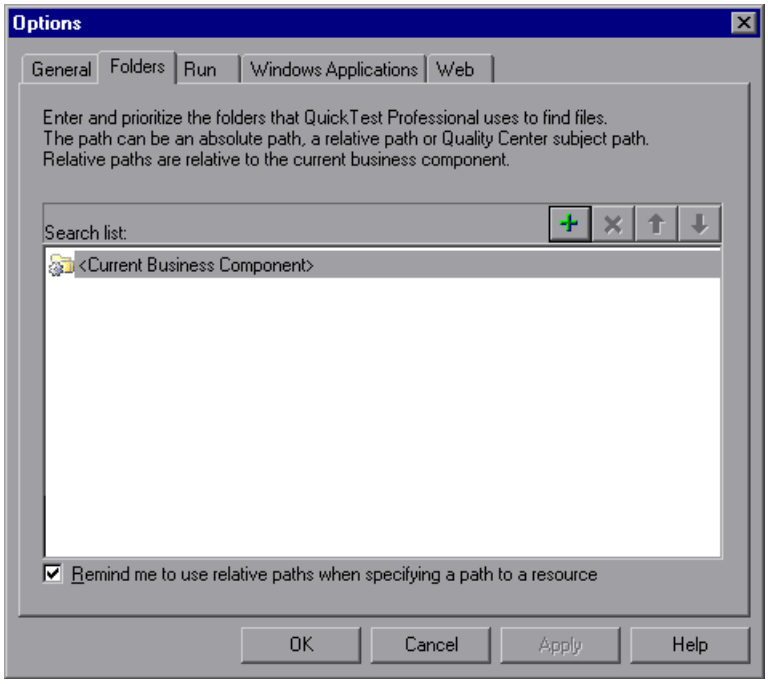
---

**Notes:**

- The current component is listed in the **Search list** by default. It cannot be deleted.
- For more information on relative or absolute paths, see the section on using relative paths in the *HP QuickTest Professional User's Guide*.

---

QuickTest searches for the specified component or file in the order in which the folders are displayed in the search list. If the same file name exists in more than one folder, QuickTest uses the first instance it finds.

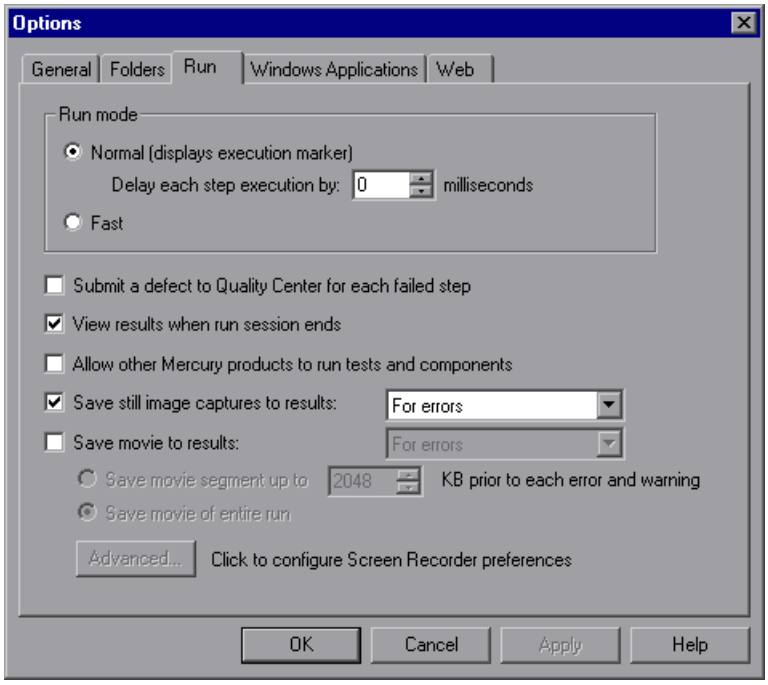


The Folders tab includes the following options:

| Option | Description |
|---|---|
| **Search list** | Indicates the folders in which QuickTest searches for components or files. If you define folders here, you do not need to designate the full path of a component or file in other dialog boxes. The order of the search paths in the list determines the order in which QuickTest searches for a specified file. |
|  | Adds a new folder to the search list.<br>**Tips:**<br>➤ To add a Quality Center path when connected to Quality Center, click this button. QuickTest adds [QualityCenter], and displays a browse button so that you can locate the Quality Center path.<br>➤ When not connected to Quality Center, hold the SHIFT key and click this button. QuickTest adds [QualityCenter], and you enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests.<br>➤ Note that QuickTest searches Quality Center project folders only when you are connected to the corresponding Quality Center project. |
|  | Deletes the selected folder from the search list. |
|  | Moves the selected folder up in the list. |
|  | Moves the selected folder down in the list. |
| **Remind me to use relative paths when specifying a path to a resource** | When saving a resource, you can choose to be prompted to use a relative path. For more information, see the section on using relative paths in the *HP QuickTest Professional User's Guide*. |

# Setting Run Testing Options

The Run tab options affect how QuickTest runs components and displays run session results in the Test Results window.



The Run tab includes the following options:

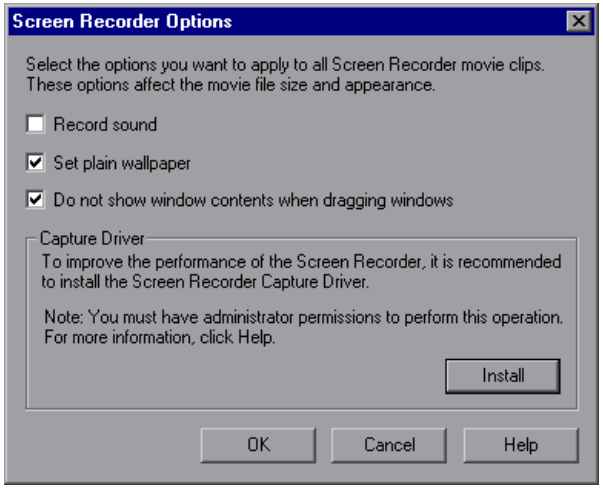
| Option | Description |
|---|---|
| **Run mode** | Instructs QuickTest how to run your component:<br>➤ **Normal (displays execution marker).** Runs your component with the execution arrow to the left of the Keyword View, marking each step as it is performed.<br>**Delay each step execution by.** You can specify the time in milliseconds that QuickTest should wait before running each consecutive step (up to a maximum of 10000 ms.)<br>The **Normal** run mode option requires more system resources than the **Fast** option, described below.<br>**Note:** You must have Microsoft Script Debugger installed to enable this mode. For more information, see the *HP QuickTest Professional Installation Guide*.<br>➤ **Fast.** Runs your component without the execution arrow to the left of the Keyword View. This option requires fewer system resources.<br>**Note:** When running a test set from Quality Center, tests are automatically run in **Fast** mode, even if **Normal** mode is selected. |
| **Submit a defect to Quality Center for each failed step** | Relevant only for tests. |
| **View results when run session ends** | Instructs QuickTest to display the results automatically following the run session. |
| **Allow other HP products to run tests and components** | Enables other HP products such as Quality Center to run QuickTest components.<br>**Note:** This option is not required to enable WinRunner to run QuickTest components. |

| Option | Description |
|---|---|
| **Save still image captures to results** | Instructs QuickTest when to capture and save still images of the application during the run session to display them in the test results.<br>Choose an option from the list:<br>➤ **Always.** Captures images for all steps in the run.<br>➤ **For errors.** Captures images only for failed steps. This is the default setting.<br>➤ **For errors and warnings.** Captures images only for steps that return a failed or warning status.<br>For more information, see "Capturing and Viewing Still Images and Movies of Your Application" on page 640. |
| **Save movie to results** | Instructs QuickTest when to capture and save a movie of the application during the run session to display it in the test results. This option is disabled by default.<br>Choose an option from the list:<br>➤ **Always.** Captures a movie of all steps in the run.<br>➤ **For errors.** Captures movies only for failed steps.<br>➤ **For errors and warnings.** Captures movies only for steps that return a failed or warning status.<br>For more information, see "Capturing and Viewing Still Images and Movies of Your Application" on page 640. |

| Option | Description |
|---|---|
| **Save movie segment up to __ KB prior to each error and warning**<br>(Enabled only when **For errors** or **For errors and warnings** is selected in the **Save movie to results** option.) | When selected, QuickTest saves movie segments for each error (or warning). Each segment contains the specified number of kilobytes of the movie prior to the failed (or warning) step. You can enter any value from 400 (0.4 MB) to 2097152 (2 GB). If more than one segment is captured for a test run, QuickTest stores a single movie with the test results that is comprised of all the relevant movie segments. |
| **Save movie of entire run**<br>(Enabled only when **For errors** or **For errors and warnings** is selected in the **Save movie to results** option.) | When selected, QuickTest saves a movie of the entire run if at least one error (or warning) occurs. |
| **Advanced**<br>(Enabled only when **Save movie to results** is selected.) | Provides advanced options for the Screen Recorder that affect the movie file size and appearance. |

## Understanding the Screen Recorder Options Dialog Box

The Screen Recorder Options dialog box enables you to set options for the Screen Recorder that affect the movie file size, appearance, and recording performance. Any settings that affect Windows functionality are restored when the session is completed.



The Screen Recorder Options dialog box includes the following options:

| Option | Description |
|---|---|
| **Record sound** | Instructs QuickTest 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** | Installs or uninstalls the Screen Recorder Capture Driver. The Screen Recorder Capture Driver improves the performance of the Screen Recorder during movie recording.<br>**Note:** The Screen Recorder Capture Driver cannot be installed or uninstalled when running QuickTest via a remote connection. |

---

**Note for Vista users:** In addition to the options described above, if your Vista Windows color scheme is set to **Aero**, QuickTest automatically sets it to **Vista Basic** while capturing movies of a run session to maximize performance. The color scheme is returned to its previous settings when the run session ends.

---

For information on working with captured movies, see "Viewing Movies of Your Run Session" on page 642.

# 21

# Working with Business Component Settings

Before you create or debug a business component, you can use the Business Component Settings dialog box to view the settings already defined for the component in its associated application area. You can also define some additional settings for the component in the Business Component Settings dialog box.

**This chapter includes:**



---

**Note:** For more information on defining component settings in application areas, see Chapter 12, "Working with Application Areas."

---

# About Working with Business Component Settings

When you create a new application area, you define the settings and resources needed to create a new business component. The settings include associated add-ins, the Windows-based applications on which the components can record and run, and the location of any function libraries and shared object repositories to use with the components.

When you (or a Subject Matter Expert) create a new component, the component is automatically linked to the settings defined in its associated application area. The Business Component Settings dialog box displays these settings in read-only format.

You can define some additional settings, such as input and output parameters, and the component status, in the Business Component Settings dialog box.

---

**Note:** You can also set testing options that affect all components. For more information, see Chapter 20, "Setting Global Testing Options."

---

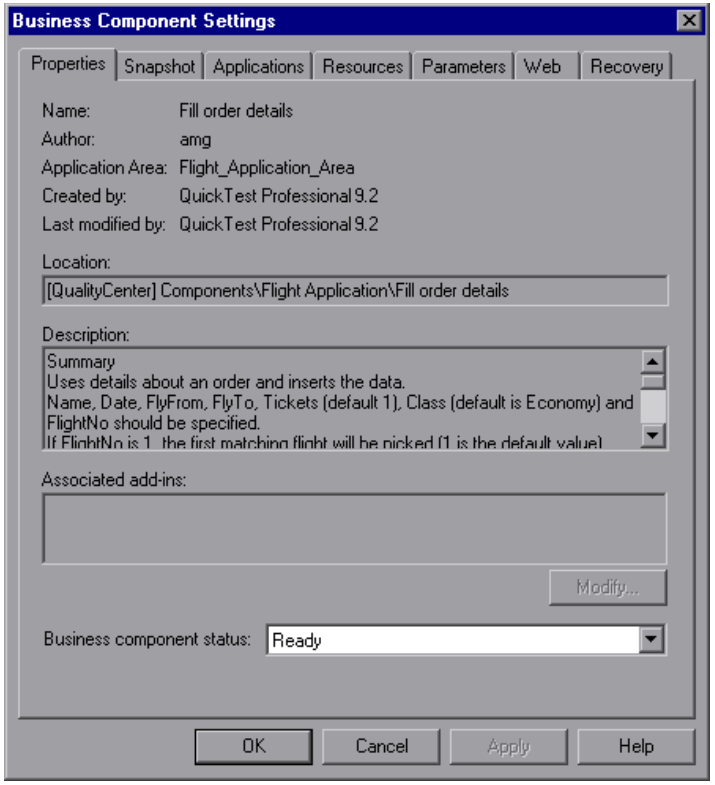
# Using the Business Component Settings Dialog Box

The Business Component Settings dialog box enables you to view settings and define specific options for a component.

**To open the Business Component Settings dialog box:**

**1** Open the component whose settings you want to view or define.

**2** Choose **File** > **Settings**, or click the **Settings** toolbar button. The Business Component Settings dialog box opens. It is divided by subject into tabbed pages.




**3** Select the required tab to view or set the options as required. See the table below for more information on the available settings and options in each tab.

**4** Click **Apply** to apply your changes and keep the dialog box open, or click **OK** to save your changes and close the dialog box.

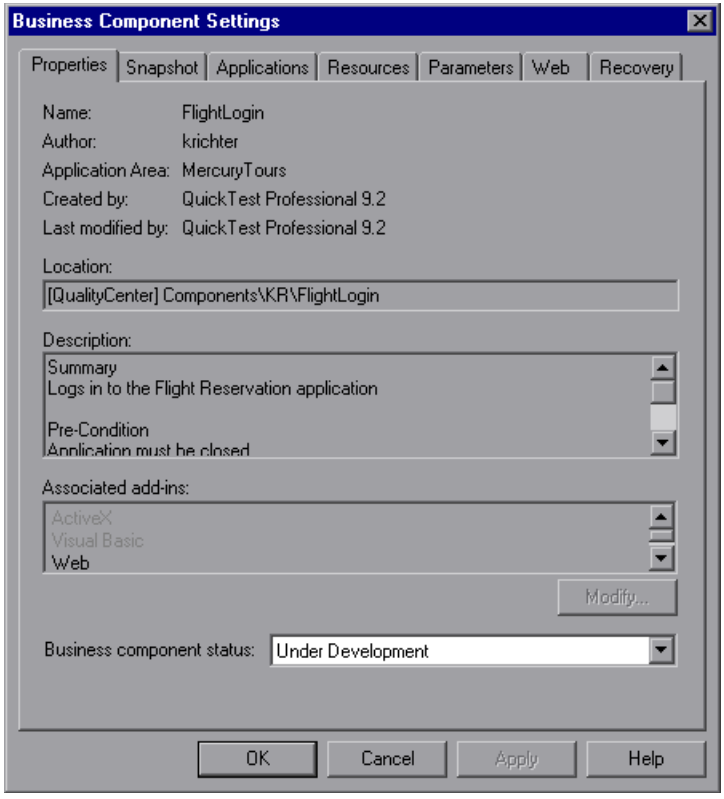
The Business Component Settings dialog box contains the following tabs:

| Tab Heading | Tab Contents |
|---|---|
| **Properties** | The properties of the business component, for example, its description and associated add-ins. You can also set the status of the component. For more information, see "Working with Component Properties" on page 601. |
| **Snapshot** | Options for capturing or loading a snapshot image to be saved with the component for display in Quality Center. For more information, see "Defining a Snapshot for Your Component" on page 604. |
| **Applications** | The Windows-based applications on which the component can record and run. For more information, see "Viewing Application Settings" on page 606. |
| **Resources** | The resources associated with the component, including the location of any function libraries and the shared object repository. For more information, see "Viewing Component Resources" on page 608. |
| **Parameters** | Options for specifying input and output parameters for the component. For more information, see "Defining Parameters for Your Component" on page 609. |
| **Recovery** | How the component recovers from unexpected events and errors that occur in your testing environment during a run session. For more information, see "Viewing Recovery Scenario Settings" on page 613. |

In addition to these tabs, the Business Component Settings dialog box may contain additional tabs for scripted components. For information on these tabs, refer to the *HP QuickTest Professional User's Guide*. There may also be other tabs depending on the add-ins that are currently loaded, for example, SAP or Web Services. For information on tabs related to add-ins, see the relevant section of the *HP QuickTest Professional Add-ins Guide*.

# Working with Component Properties

You can use the Properties tab of the Business Component Settings dialog box to view general information about your component, including its description and any add-ins associated with it. You can also set or modify its status.



The Properties tab includes the following items:

| Setting | Description |
|---|---|
| **Name** | Indicates the name of the component. You assign a name to the component when you save it. |
| **Author** | Indicates the Windows user name of the person who created the component. |
| **Application Area** | Indicates the name of the application area which is associated with the component. For more information, see "Creating a New Business Component" on page 458.<br>**Note:** If the component was created in Quality Center 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 by** | Indicates the version of QuickTest used to create the test. |
| **Last modified by** | Indicates the version of QuickTest last used to modify the test. |
| **Location** | Indicates the Quality Center path and filename of the component.<br>**Note:** If the component is not yet saved, the location indicates **Not saved**. |
| **Description** | Displays the description specified for your component. This field can be entered or modified only in Quality Center. |
| **Associated add-ins** | Displays 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. |
| **Business Component Status** | Specifies the status of the component. You can change the status of the component by selecting a different option from the list. For more information on status options, see "Understanding Component Statuses" on page 603. |

For information on defining general information for the application area on which your component is based, see "Defining General Settings" on page 421.

## Understanding Component Statuses

Business components can be assigned statuses either in QuickTest or in Quality Center. A business component status can either be manually specified, or in certain cases may be automatically assigned by Quality Center. 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.

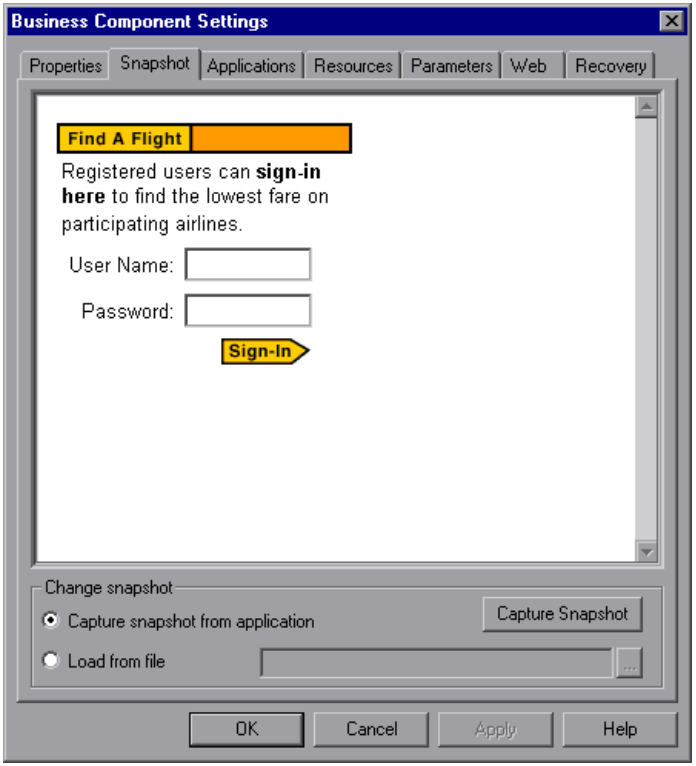
A component can be assigned one of the following statuses:

- **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 Quality Center with Business Process Testing support.
  - Component requests dragged into the component tree in Quality Center with Business Process Testing support.

➤ **Not Implemented.** The component has been requested in the Test Plan module of Quality Center. 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.

## Defining a Snapshot for Your Component

The Snapshot tab of the Business Component Settings dialog box 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. The Subject Matter Expert can view the image in Quality Center, in the component and in any business process test in which the component is included.



**Note:** The snapshot image can also be captured and saved with the component from the Snapshot tab in Quality Center when installed with Business Process Testing support. For information on capturing a snapshot for a component in Quality Center, refer to the *HP Business Process Testing User's Guide*.

The Snapshot tab contains the following options:

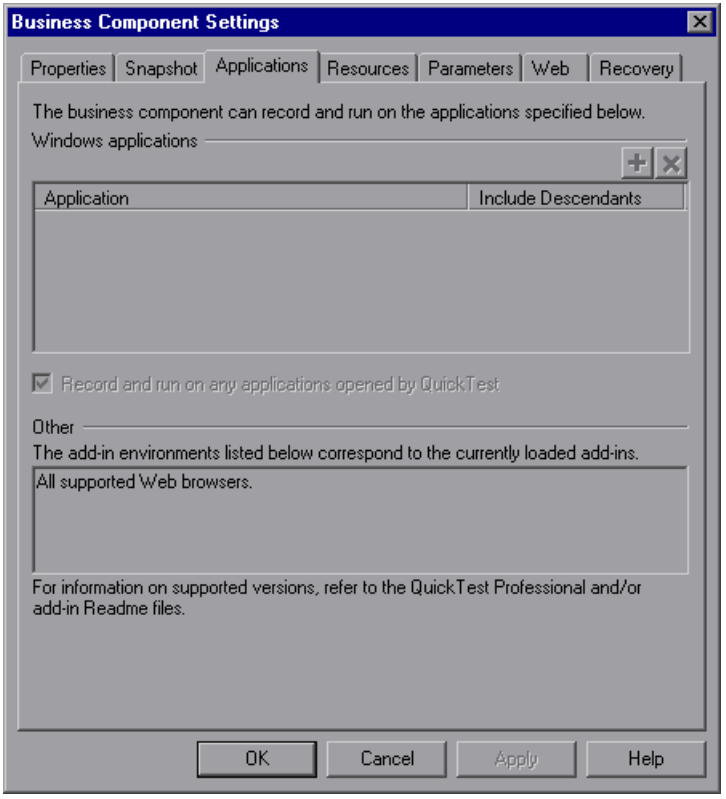
| Option | Description |
|---|---|
| **Capture snapshot from application** | Enables you to define the image to be captured by clicking the **Capture Snapshot** button. You can then drag the crosshairs pointer to select the area to be captured. When you release the mouse button, the captured area is 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. |

When you click **Apply** or **OK**, the image is saved with the component and is displayed in the business process tests containing this component in Quality Center.

# Viewing Application Settings

The Applications tab of the Business Component Settings dialog box displays a read-only list of the Windows-based applications on which the component can record and run (based on its current application area). You can record steps only on the specified applications.

It also displays the environments on which the component can currently record (based on the currently loaded add-ins).



You specify the Windows-based applications on which the component can record and run in the associated application area settings. For more information, see "Defining Application Settings for Your Application Area" on page 443.

---

**Notes:**

- If you are recording a new component and have not yet set your application settings in the Applications tab of the Application Area Settings dialog box, the Applications dialog box opens when you start to record. The Applications dialog box contains the same options as the Applications tab, described in "Defining Application Settings for Your Application Area" on page 443.
- The Applications dialog box and Applications tab may also contain options applicable to any QuickTest add-ins installed on your computer. For information regarding these options, refer to the documentation provided for the specific add-in.

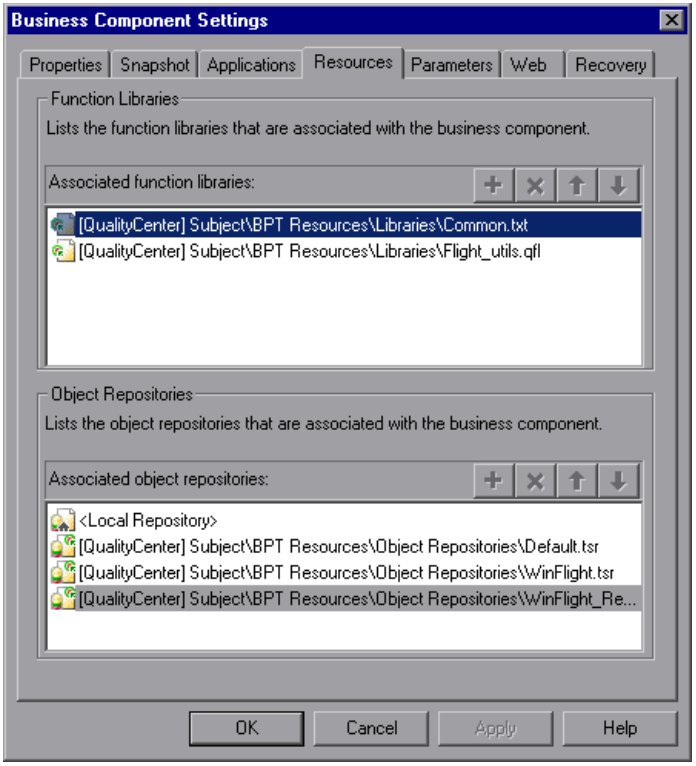
---

The Applications tab includes the following items:

| Setting | Description |
|---|---|
| **Application** | Lists the details of the applications on which to record and run the component.<br>The application list is left blank if you do not want to record or run on Windows applications. (This is the default setting.) |
| **Record and run on any applications opened by QuickTest** | Records and runs on any applications invoked by QuickTest (as child processes of QuickTest). For example, applications opened during a record or run session using an OpenApp function, or another operation containing a function that opens an application. |
| **Other** | Lists the add-in environments that correspond to the currently loaded add-ins. |

# Viewing Component Resources

The Resources tab of the Business Component Settings dialog box displays a read-only list of the function libraries and object repositories associated with your component (via its associated application area). All specified resources files must be saved in your Quality Center project.



The Resources tab includes the following items:

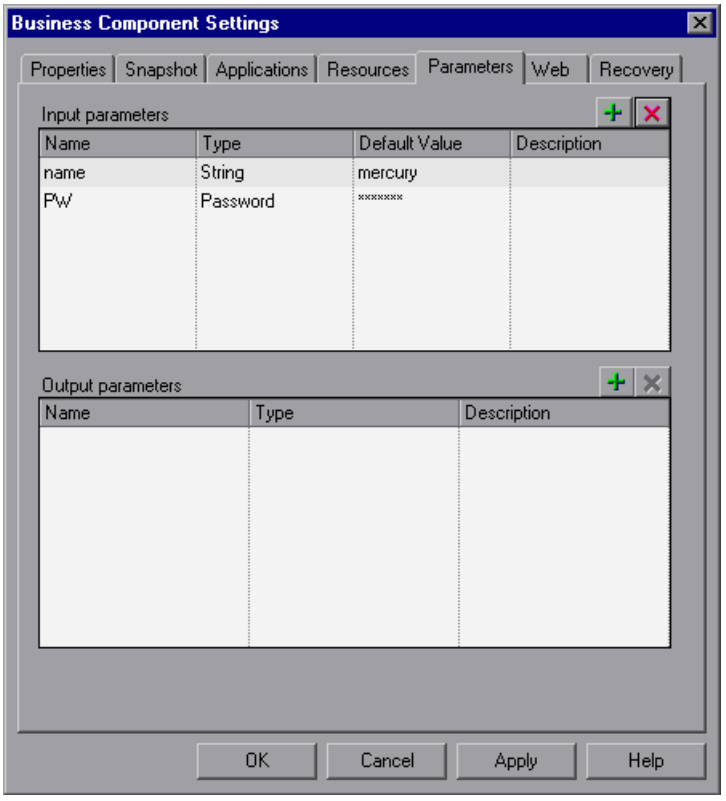
| Setting Area | Description |
|---|---|
| **Associated function libraries** | Displays the list of function libraries currently associated with your component (via its associated application area). For more information on associating function libraries, see "Managing Function Libraries" on page 426, and "Working with Associated Function Libraries" on page 387. |
| **Object Repositories** | Displays the list of shared object repositories currently associated with your component via its associated application area (in addition to the local object repository). Components use shared object repository files stored in Quality Center. For more information on associating object repositories with application areas, see "Managing Shared Object Repositories" on page 432. |

## Defining Parameters for Your Component

In the Parameters tab of the Business Component Settings dialog box, you can define input component parameters that pass values into your component and output component parameters that pass values from your component to external sources. You can also use the Parameters tab to modify or delete existing component parameters.

Component parameters are parameters that can be used to parameterize input and output values in component steps. For information on using parameter values in component steps, see the section on working with parameters in the *HP QuickTest Professional User's Guide*. For information on working with component parameters in steps, see "Using Component Parameters in Steps" on page 613.

The Subject Matter Expert can also define component parameters in Quality Center. For information, refer to the *HP Business Process Testing User's Guide*.




The Parameters tab contains two parameter lists:

- **Input parameters.** Specifies the parameters that the component can receive from the source that runs or calls it.
- **Output parameters.** Specifies the parameters that the component can pass to the source that runs or calls it.

You can edit an existing parameter by selecting it in the appropriate list and modifying its details (except for its name which cannot be modified).

---

**Note:** The input and output parameter lists can also be modified in the Quality Center Business Components module. For more information, refer to the *HP Business Process Testing User's Guide*.

---

You can add and remove input and output parameters for your business component using the following buttons:

| Option | Description |
|---|---|
|  | Adds a parameter to the appropriate parameter list. Enter a name (case sensitive) for the new parameter and select the parameter type. Possible types are **String**, **Boolean**, **Date**, **Number**, or **Password**. You can enter a description for the parameter, for example, the purpose of the parameter in the component.<br>If you are defining an input parameter, a default value for the specified parameter type is automatically entered. You can enter or modify the default value for the parameter in the **Default Value** column. For more information, see "Defining Default Values for Input Component Parameters", below. |
|  | Removes the selected parameter from the component. |

## Defining Default Values for Input Component Parameters

When a business component runs, the actual values used for parameters are generally those sent by the application calling the component (either QuickTest or Quality Center) as described in the table below:

| Business Component Called From: | Parameter Values Specified In: |
|---|---|
| **QuickTest** | Input Parameters tab of the Run dialog box. For more information, see "Understanding the Input Parameters Tab" on page 621. |
| **Quality Center** | Component Iterations dialog box (Test Plan module). For more information, refer to the *HP Business Process Testing User's Guide*. |

If, during a component run, a value is not supplied by QuickTest or Quality Center for one or more input parameters, QuickTest uses the default value for the parameter.

When you define a new parameter in the Parameters tab of the Business Components Settings dialog box, you can specify the default value for the parameter or you can keep the default value that QuickTest assigns for the specified parameter type as follows:

| Value Type | QuickTest Default Value |
|---|---|
| **String** | Empty string |
| **Boolean** | True |
| **Date** | The current date |
| **Number** | 0 |
| **Password** | Empty string |

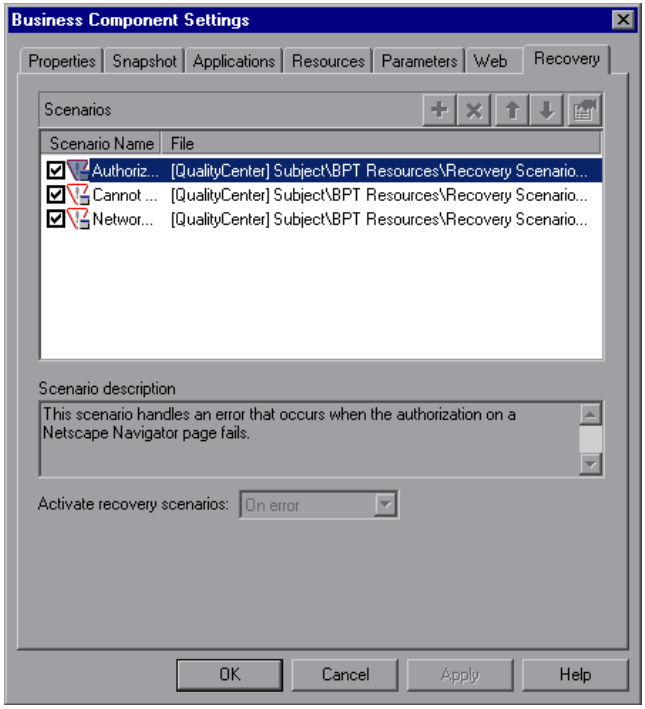
### Using Component Parameters in Steps

After you define component parameters, you can use them to parameterize values in the steps of your component by selecting input component parameters in the Value Configuration Options dialog box, or by selecting output component parameters in the Output Options dialog box. You can also use local parameters in steps. For more information on using component and local parameters in steps, see the section on working with parameters in the *HP QuickTest Professional User's Guide*.

## Viewing Recovery Scenario Settings

Recovery scenario settings enable you to specify how a business component recovers from unexpected events and errors during a run session.

The Recovery tab of the Business Component Settings dialog box displays a read-only list of all the recovery scenarios associated with the current component's associated application area.



You define the recovery scenario settings for a component in its associated application area. For more information, see "Defining Recovery Scenario Settings for Your Application Area" on page 447.

The Recovery tab includes the following items:

| Setting Area | Description |
|---|---|
| **Scenarios** | Displays the name and recovery file path for each recovery scenario associated with your component (via its associated application area). The scenario type is indicated by an icon. For more information, see "Specifying Associated Recovery Scenarios" on page 449. |
| **Scenario description** | Displays the textual description of the scenario selected in the **Scenarios** box. |
| **Activate recovery scenarios** | Displays the setting that instructs QuickTest to check whether to run the associated scenarios as follows:<br>➤ **On every step.** The recovery mechanism is activated after every step.<br>➤ **On error.** The recovery mechanism is activated only after steps that return an error return value.<br>➤ **Never.** The recovery mechanism is disabled. |

# Part VI

## Running and Analyzing Components

# 22

# Running Components

After you create a component, you can run it to check the behavior of your application.

**This chapter includes:**



## About Running Components

When you run a component, QuickTest performs the steps it contains. If you have defined component parameters, QuickTest prompts you to enter values for them. When the run session is complete, QuickTest displays a report detailing the results. For more information on viewing the results, see Chapter 23, "Viewing Run Session Results."

You can run the entire component from the beginning, or you can run part of it. You can update your component to change the test object descriptions. You can run components on objects with dynamic descriptions. For more information, see Chapter 4, "Working with Objects."

# Running Your Entire Component

QuickTest always runs a component from the first step, unless you specify otherwise. To run a component from or to a selected step you can use the **Run from Step** or **Run to Step** options. These features are useful if you want to check a specific section of the component, without running the component from the beginning or to the end. For more information, see "Running Part of Your Component" on page 622.

When you start to run a component, the Run dialog box opens, to enable you to specify the location for the results and to enter the values for any component parameters you have defined.

**To run a component:**

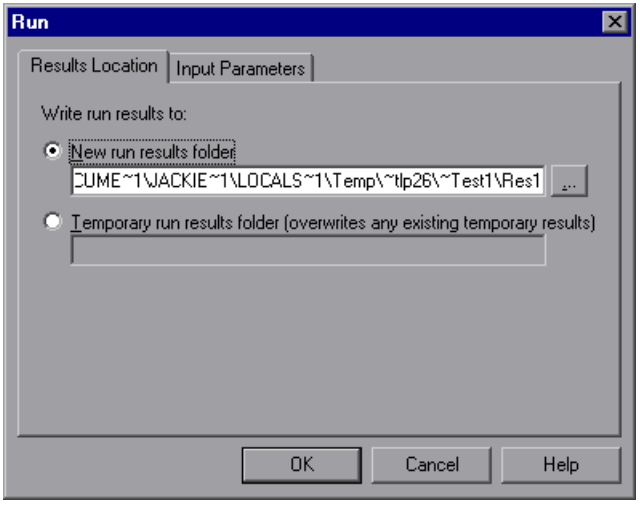
**1** If your component is not already open, choose **File** > **Open** > **Business/Scripted Component** or click the **Open** button to open it.

---

**Tip:** If you recently opened your component, you can also choose it from the recent files list in the **File** menu.

---

**2** Click the **Run** button on the toolbar, or choose **Automation** > **Run**. The Run dialog box opens.



**3** Specify the results location and the input parameter values (if applicable) for the run session. For more information, see "Understanding the Results Location Tab" on page 620, and "Understanding the Input Parameters Tab" on page 621.

**4** Click **OK**. The Run dialog box closes and the run session starts. By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 23, "Viewing Run Session Results."

---

**Note:** If you cleared the **View results when run session ends** check box in the Run tab of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 20, "Setting Global Testing Options."

---

---

**Tip:** If you want to interrupt a run session, do either of the following:

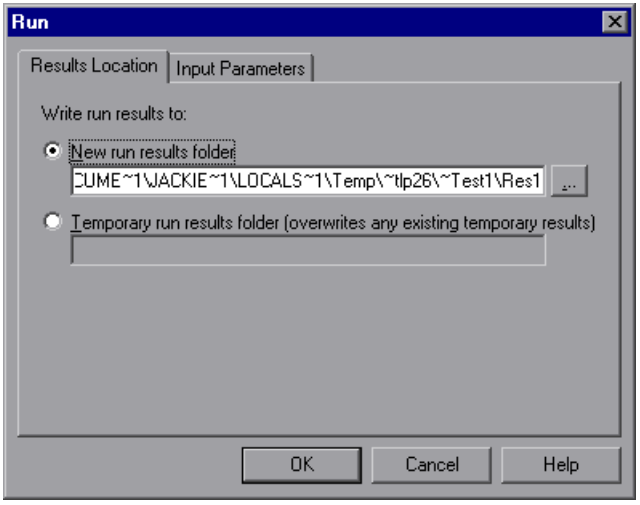
- Click the **Pause** button in the Debug toolbar or choose **Debug** > **Pause**. The run pauses. To resume running a paused run session, click the **Run** button or choose **Automation** > **Run**.
- Click the **Stop** button or choose **Automation** > **Stop**. The run session stops and the Test Results window opens.

The run session is also interrupted if you perform a file operation (for example, open a different component or create a new component).

---

## Understanding the Results Location Tab

The Results Location tab enables you to specify the location in which you want to save the run session results.




Select one of the following options:

- **New run results folder.** This option displays the default path and folder name in which the results are saved. By default, the results for components are stored in a Quality Center cache folder on your computer.

  Accept the default settings, or enter a new path by typing it in the text box or clicking the browse button to locate a different folder. The folder must be new, empty, or contain only QuickTest test or component files.

- **Temporary run results folder.** Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder.

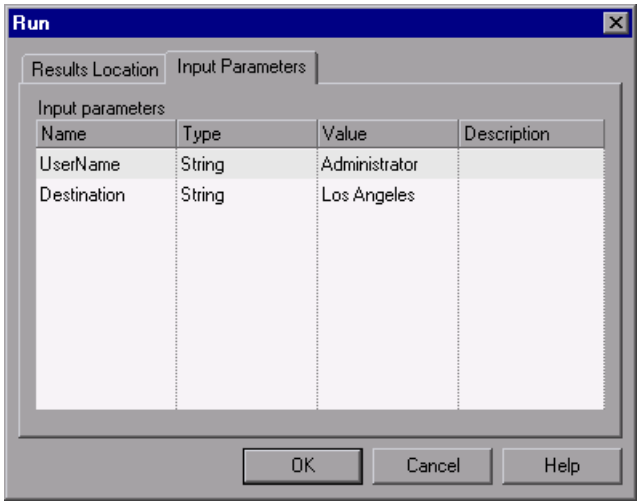
---

**Note:** QuickTest stores temporary results for all components in **<System Drive>\Documents and Settings\<user name>\Local Settings\Temp\TempResults**. The path in the text box of the **Temporary run results folder** option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts.

---

## Understanding the Input Parameters Tab

The Input Parameters tab enables you to specify the run-time values of input parameters to be used during the run session.




The Input Parameters tab displays the input parameters that were defined for the component (using the **File** > **Settings** > **Parameters** tab).

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, QuickTest uses the default value from the Business Component Settings dialog box during the run session.

For more information on setting component parameters, see "Defining Parameters for Your Component" on page 609. For more information on using parameters, see the section on working with parameters in the *HP QuickTest Professional User's Guide*.

## Running Part of Your Component

You can use the **Run from Step** option to run a selected part of your component from the selected step to the end of the component. This enables you to check a specific section of your application or to confirm that a certain part of your component runs smoothly.

---

**Note:** You can also use the **Debug** > **Run to Step** option if you want to run a component in debug mode from the start of the component to a selected step. For more information, see "Using the Run to Step and Debug from Step Commands" on page 687.

---

**To run a component from a selected step:**

1. Open your application to the location matching the step you want to run.
2. Select the step where you want to start running the component.

   Make sure that the step you choose is not dependent on previous steps.
3. Choose **Automation** > **Run from Step**.
4. In the Run dialog box, choose where to save the run session results, and any input parameters you want to use, as described in "Understanding the Results Location Tab" on page 620, and "Understanding the Input Parameters Tab" on page 621.
5. Click **OK**. The Run dialog box closes and the run session starts.

By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 23, “Viewing Run Session Results.”

The Test Results summary displays a note indicating that the component was run using the **Run from Step** option.

---

**Note:** If you cleared the **View results when run session ends** check box in the Run tab of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 20, “Setting Global Testing Options.”

---

# 23

# Viewing Run Session Results

After running a component, you can view a report of major events that occurred during the run session.

---

**Note:** You cannot view business process test run results when you open the Test Results window from QuickTest. To view run results for a business process test, select the results for the iteration you want to view and open them from within Quality Center.

---

**This chapter includes:**

# About Viewing Run Session Results

When a run session ends, you can view the run session results in the Test Results window. By default, the Test Results window opens automatically at the end of a run. If you want to change this behavior, clear the **View results when run session ends** check box in the Run tab of the Options dialog box.

The Test Results window contains a description of the steps performed during the run session. It displays a single run iteration.

After you run a component, the Test Results window displays all aspects of the run session and can include:

- a high-level results overview report (pass/fail status)
- the data used in all runs
- an expandable tree of the steps, specifying exactly where application failures occurred
- the exact locations in the component where failures occurred
- a still image of the state of your application at a particular step
- a movie clip of the state of your application at a particular step or of the entire component
- detailed explanations of each step pass or failure, at each stage of the component

---

**Note:** The Test Results window can show results with up to 300 levels in the tree hierarchy. If you have results with more than 300 nested levels, you can view the entire report by manually opening the **results.xml** file.

---

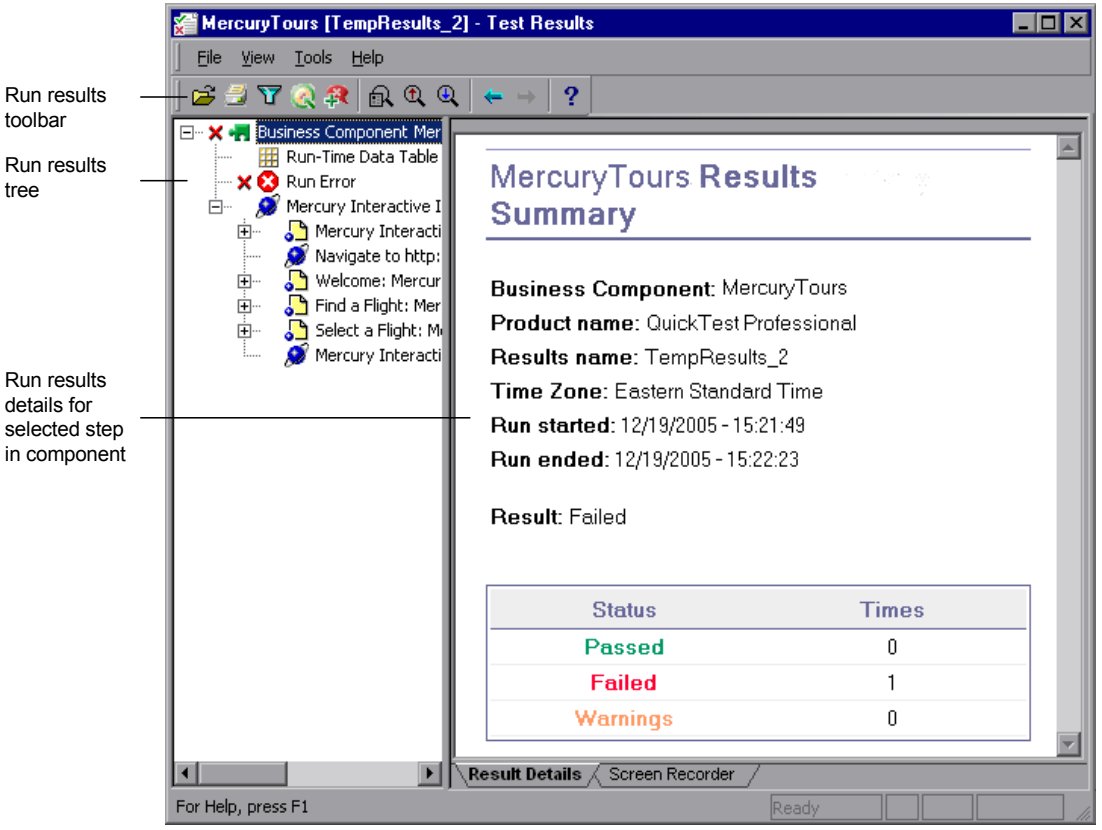
# The Test Results Window

After a run session, you view the results in the Test Results window. By default, the Test Results window opens when a run session is completed. For information on changing the default setting, see "Setting Run Testing Options" on page 591.

The left pane of the Test Results window contains the run results tree. The right pane of the Test Results window contains the details for a selected step in the run results tree. The details for a selected step may include a component summary, step details, a still image of your application, or a movie of your application.

You can open the Test Results window as a standalone application from the **Start** menu. To open the Test Results window, choose **Start** > **Programs** > **QuickTest Professional** > **Test Results Viewer**.

Below is an example of the run results for a component:




---

**Note:** In this example, the component failed due to a run error in an associated function library. If the run error had not occurred, the **Result** would indicate **Done**.

---

The Test Results window contains the following key elements:

- **Test results title bar.** Displays the name of the component.
- **Menu bar.** Displays menus of available commands.
- **Run results toolbar.** Contains buttons for viewing run session results (choose **View** > **Test Results Toolbar** to display the toolbar). For more information, see "Run Results Toolbar" on page 632.
- **Run results tree.** Contains a graphic representation of the run results in the run results tree. The run results tree is located in the left pane of the Test Results window. For more information, see "Run Results Tree" on page 629.
- **Result Details tab.** Contains details of the selected node in the run results tree. The Result Details tab is located in the right pane of the Test Results window. For more information, see "Run Results Details" on page 630.
- **Screen Recorder tab.** Contains the recorded movie associated with the test results. The screen recorder tab is located in the right pane of the Test Results window. For more information, see "Capturing and Viewing Still Images and Movies of Your Application" on page 640.
- **Status bar.** Displays the status of the currently selected command (choose **View** > **Status Bar** to view the status bar).

You can change the appearance of the Test Results window. For more information, see "Changing the Appearance of the Test Results Window" on page 632.

## Run Results Tree

The left pane in the Test Results window displays the **run results tree**—a graphical representation of the run session results:

- ✔ indicates a step that succeeded. This icon is displayed only if the component step contains one of the following:
  - Verify operations (functions), such as VerifyProperty
  - AddToTestResults (or its equivalent) with a **micPass** status
- ✘ indicates a step that failed. Note that this causes all parent steps (up to the root component) to fail as well.

- ! indicates a warning, meaning that the step did not succeed, but it did not cause the component to fail.
- ! indicates a step that failed unexpectedly, such as when an object is not found for a checkpoint.
- indicates that the Smart Identification mechanism successfully found the object.
- indicates that a recovery scenario was activated.
- indicates that the run session was stopped before it ended.
- [ password ].SetSecure square brackets around a test object name indicate that the test object was created dynamically during the run session. A dynamic test object is created either using programmatic descriptions or by using an object returned by a ChildObjects method, and is not saved in the object repository.
- displays the Maintenance Mode Update Result, a table that describes the **Action** taken by Maintenance Run Wizard on a failed step, and the **Details** of that action. Displayed only for components run in Maintenance Run Mode. For more information on Maintenance Run Mode, see Chapter 26, "Maintaining Components."

  You can collapse or expand a branch in the run results tree to change the level of detail that the tree displays.

## Run Results Details

By default, when the Test Results window opens, a component summary is displayed in the **Result Details** tab in the right pane of the window.

The right pane of the Test Results Window contains tabs labeled **Result Details** and **Screen Recorder**. When you select the top node of the run results tree, the Result Details tab contains a summary of the results for your component. When you select a branch or step in the tree, the Result Details tab contains the details for that step. The Result Details tab may also include a still image of your application for the highlighted step.

The Screen Recorder tab contains the movie associated with your test results. If there is no movie associated with your test results, the Screen Recorder tab contains the message: No movie is associated with the results.

For more information on viewing still images and movies of your application, see "Capturing and Viewing Still Images and Movies of Your Application" on page 640.

When you select the top node of the run results Tree, the Result Details tab indicates the component name, product name (for a component), results name, the start and end date and time of the run session, and whether an iteration passed or failed. For a component, the possible results are **Done** or **Failed**.

In addition, if the Web Services Add-in is installed and was loaded during the run session, the Web Services run toolkit is displayed in the Result Details tab. The run toolkit is displayed even if the component does not include any Web Services steps.

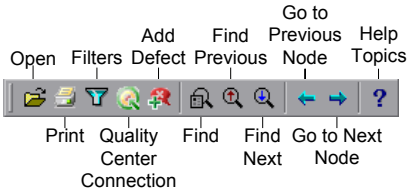
If the component was run in **Maintenance Run Mode**, the Results Details tab contains a **Maintenance Summary**. The **Maintenance Summary** lists the number of objects that were updated and added in your component. It also lists the number of updated and commented steps in your component. The **Object Repository Changes Report** lists the specific changes that the Maintenance Run Wizard made to the object repository and contains the following sections:

- **Added Objects.** Lists the objects that were added to the object repository by the Maintenance Run Wizard.
- **Object with Changed Descriptions.** Describes the changes to object properties carried out by the Maintenance Run Wizard.

For more information on Maintenance Run Mode, see "Maintaining Components" on page 701.

## Run Results Toolbar

The Run Results toolbar contains buttons for viewing test results.




## Changing the Appearance of the Test Results Window

By default, the Test Results window has the same look and feel as the QuickTest window, using the Microsoft Office 2003 theme. You can change the look and feel of the Test Results window, as required.

**To change the appearance of the Test Results window:**

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

---

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

---

---

**Tip:** You can also change the theme used for the main QuickTest window. For more information, see "Changing the Appearance of the QuickTest Window" on page 54.

---

# Viewing the Results of a Run Session

By default, at the end of the run session, the results are displayed in the Test Results window. (You can change the default setting in the Options dialog box. For more information, see "Setting Run Testing Options" on page 591.)

In addition, you can view the results of previous runs of the current component, and results of other components. You can also preview run session results on screen and print them to your default Windows printer, as well as export them to an HTML file.

For more information, see:

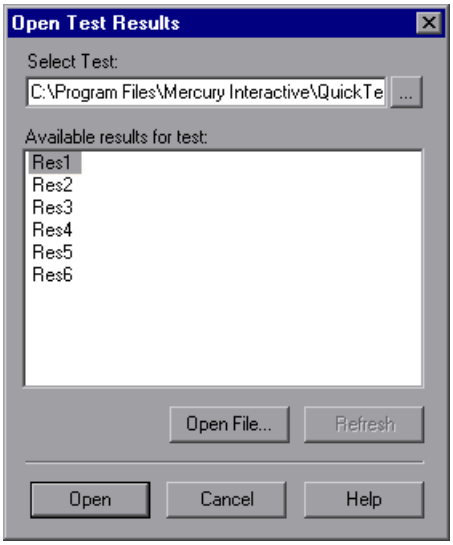

- "Opening Test Results to View a Selected Run" on page 634
- "Working with the Test Results Window" on page 636
- "Viewing Results of Components Run From Quality Center" on page 640
- "Capturing and Viewing Still Images and Movies of Your Application" on page 640
- "Finding Results Steps" on page 645
- "Printing Run Session Results" on page 646
- "Previewing Test Results" on page 647
- "Exporting Test Results" on page 649

## Opening Test Results to View a Selected Run

You can view the saved results for the current component, or you can view the saved results for other components.

You select the run results to open for viewing from the Open Test Results dialog box, which opens when:

- You choose **File** > **Open** from within the Test Results window.
- 
  You click the **Results** button in the QuickTest window or choose **Automation** > **Results**, when there are several results, or no results, for the current component.




The results of run sessions for the current component are listed. To view one of the results sets, select it and click **Open**.

---

**Tip:** To update the results list after you change the specified component path, click **Refresh**.

---

To view results of runs for other components, you can search by component result file.

---

**Note:** You cannot view business process test run results when you open the Test Results window from QuickTest. To view run results for a business process test, select the results for the iteration you want to view and open them from within Quality Center.

---

## Searching for Results in the File System

By default, the results for component runs are stored in a Quality Center cache folder on your computer. When you run your component, you can specify a different location to store the results, using the Results Location tab of the Run dialog box. Specifying your own location for the results file can make it easier for you to locate the results file in the file system. For more information, see "Understanding the Results Location Tab" on page 620.

You can search for results in the file system by component or by result file.

**To search for results in the file system by component:**

1. In the Open Test Results dialog box, enter the path of the folder that contains the results file for your component, or click the browse button to open the Open Test dialog box.
2. Find and highlight the component whose results you want to view, and click **Open**.
3. In the Open Test Results dialog box, highlight the component result set you want to view, and click **Open**. The Test Results window displays the selected results.

**To search for results in the file system by result file:**

1. In the Open Test Results dialog box, click the **Open File** button to open the Select Results File dialog box.
2. Browse to the folder where the component results file is stored.

**3** Highlight the results (**.xml**) file you want to view, and click **Open**. The Test Results window displays the selected results.

## Working with the Test Results Window

The Test Results window contains a graphic and text summary of the results of a run as well as details of each step in the run.

**To view the results of a run:**

**1** If the Test Results window is not already open, click the **Results** button or choose **Automation** > **Results**.

---

**Tip:** You can open the Test Results window as a standalone application from the **Start** menu. To open the Test Results window, choose **Start** > **Programs** > **QuickTest Professional** > **Test Results Viewer**.

---

- If there are run session results for the current component, they are displayed in the Test Results window. For information on the Test Results window, see "The Test Results Window" on page 627.
- If there are no run session results for the current component, the Open Test Results dialog box opens. You can select the run session results for any component, or you can search for the run session results (**results.xml**) file anywhere in the file system. Click **Open** to display the selected results in the Test Results window. For more information on viewing run session results, see "Viewing the Results of a Run Session" on page 633.

**2** You can collapse or expand a branch in the run results Tree to select the level of detail that the tree displays.

- To collapse a branch, select it and click the collapse (–) sign to the left of the branch icon, or press the minus key (–) on your keyboard number pad. The details for the branch disappear in the results tree, and the collapse sign changes to expand (+).
- To collapse all of the branches in the run results tree, choose **View** > **Collapse All** or right click a branch and select **Collapse All**.

➤ To expand a branch, select it and click the expand (+) sign to the left of the branch icon, or press the plus key (+) on your keyboard number pad. The tree displays the details for the branch and the expand sign changes to collapse.

If you just opened the Test Results window, the tree expands one level at a time. If the tree was previously expanded, it reverts to its former state.

➤ To expand a branch and all branches below it, select the branch and press the asterisk (*) key on your keyboard number pad.

➤ To expand all of the branches in the run results tree, choose **View** > **Expand All**; right click a branch and select **Expand All**; or select the top level of the tree and press the asterisk (*) key on your keyboard number pad.

**3** You can view the results of an iteration or a step. When you select a step in the run results tree, the right side of the Test Results window contains the details of the selected step. Depending on your settings in the Run tab of the Options dialog box, the right side of the Test Results window may be split into two panes, with the bottom pane containing a still image (or in selected cases, other data) of the selected step. The right pane may also contain a movie of your application. For more information, see "Capturing and Viewing Still Images and Movies of Your Application" on page 640 and "Setting Run Testing Options" on page 591.

The results can be one of the following types:

➤ Steps that were not successful, but did not cause the component to stop running, are marked **Warning** in the right part of the Test Results window and are identified by the icon  or .

---

**Note:** A component containing a step marked **Warning** may still be labeled **Done**.

---

**4** To filter the information displayed in the Test Results window, click the **Filters** button or choose **View** > **Filters**. The Filters dialog box opens.


The default filter options are displayed in the image above. The Filters dialog box contains the following options:

**Status** area:

- **Fail.** Displays the run results for the steps that failed.
- **Warning.** Displays the run results for the steps with the status **Warning** (steps that did not pass, but did not cause the component to fail).
- **Pass.** Displays the run results for the steps that passed.
- **Done.** Displays the run results for the steps with the status **Done** (steps that were performed successfully but did not receive a pass, fail, or warning status).

---

**Note:** The **Iterations** and **Content** areas are not relevant for components.

---

**5** To find specific steps within the Test Results, click the **Find** button or choose **Tools** > **Find**. For more information, see "Finding Results Steps."

**6** To move between previously selected nodes within the run results tree, click the **Go to Previous Node** or **Go to Next Node** buttons.

**7** To view the results of other run sessions, click the **Open** button or choose **File** > **Open.** For more information, see "Opening Test Results to View a Selected Run" on page 634.

**8** To print run results, click the **Print** button or choose **File** > **Print**. For more information, see "Printing Run Session Results" on page 646.
(You can preview the run results before you print them. For more information, see "Previewing Test Results" on page 647.)

---

**Note:** If you have Quality Center installed, you can add a defect to a Quality Center project. For more information, see "Manually Submitting Defects Detected During a Run Session to a Quality Center Project" on page 660.

---

**9** To export the run results to an HTML file, choose **File** > **Export to HTML File**. For more information, see "Exporting Test Results" on page 649.

**10** Choose **File** > **Exit** to close the Test Results window.

## Viewing Results of Components Run From Quality Center

When you run business process tests containing QuickTest components from Quality Center, the Quality Center server opens QuickTest on the host computer and runs the components from that computer. All run results are then saved to the default location for those components.

You can view the results of QuickTest components run from Quality Center. If your results include a movie of your application, the movie can be viewed in Quality Center. The run results contain the same information described in "The Test Results Window" on page 627 plus the following additional fields:

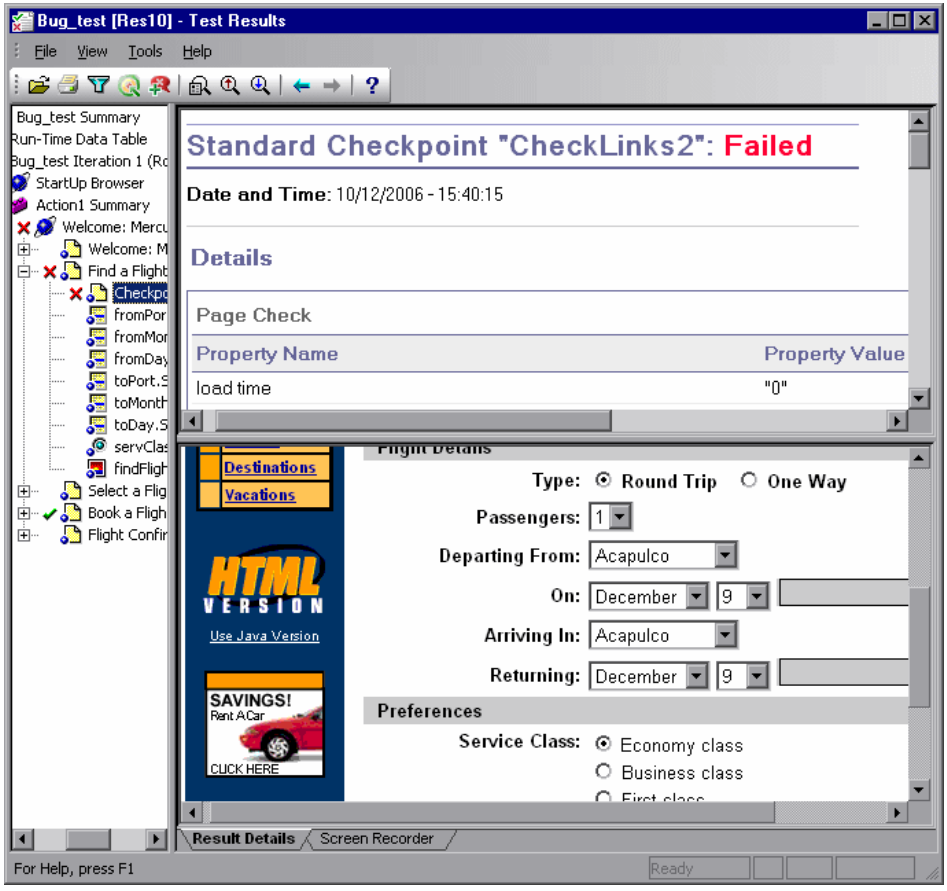
- **Test set.** Specifies the location of the business process test.
- **Test instance.** Specifies the instance number of the test in the business process test. For example, if the same test is included twice in the business process test, you can view the results of Test instance 1 and Test instance 2.

## Capturing and Viewing Still Images and Movies of Your Application

QuickTest Professional can capture still images and movies of your application during a run session. These captured files can be viewed in the Test Results window. The right pane of the Test Results window contains tabs labeled **Result Details** and **Screen Recorder**. These tabs enable you to view either still images and text details, or a movie of your application.

## Viewing Still Images of Your Application

By default, QuickTest saves a still image of your application for failed steps. When you select a failed step in the run results tree and select the **Result Details** tab, the bottom right pane of the Test Results window displays a screen capture of your application corresponding to the highlighted step in the run results tree.



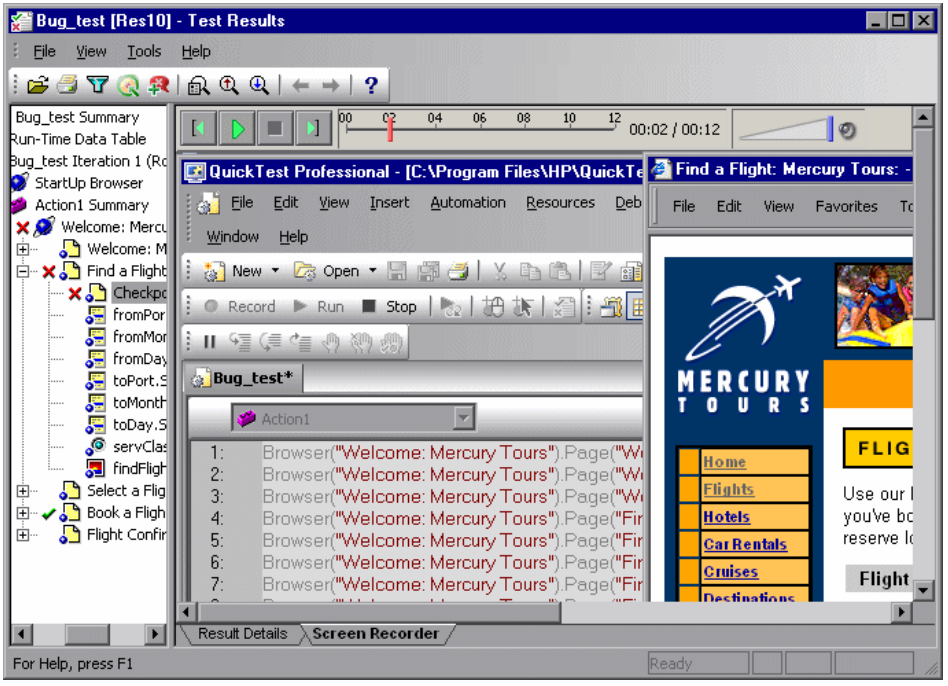

If the highlighted step does not contain an error, the right pane contains the result details with no screen capture.

You can customize the criteria QuickTest uses to save still images by selecting **Always**, **For errors**, or **For errors and warnings** in the **Save still image captures to results** list in the Run tab of the Options dialog box. For more information, see "Setting Run Testing Options" on page 591.

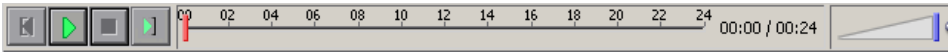
## Viewing Movies of Your Run Session

QuickTest can save a movie of your application during a run session. This can be useful to help you see how your application behaved under test conditions or to debug your component. You can view the entire movie or select a particular segment to view. When you select a step in the run results tree and click the **Screen Recorder** tab, the right pane of the Test Results window displays the frame in the movie corresponding to the highlighted step in the run results tree.

You can customize the criteria QuickTest uses to save movies by selecting **Always**, **For errors**, or **For errors and warnings** in the **Save movies to results** list in the Run tab of the Options dialog box. For more information, see "Setting Run Testing Options" on page 591.



The top of the Screen Recorder tab contains controls that enable you to play, pause, stop, jump to the first frame of the movie, jump to the last frame of the movie, and control the volume. You can also drag the slider bar to scroll through the movie.




---

**Tips:**

- You can double-click the right pane of the Test Results window to expand the Screen Recorder and hide the run results tree. Double-clicking again restores the Screen Recorder to its previous size and displays the run results tree. When the Screen Recorder is expanded, the playback controls at the top of the Screen Recorder automatically hides after approximately three seconds with no mouse activity, or when you click anywhere on the Screen Recorder. They reappear when you move the mouse again.
- The Screen Recorder saves a movie of your entire desktop. You can prevent the QuickTest window from partially obscuring your application while capturing the movie by minimizing QuickTest during the run session. For information on how to minimize QuickTest during run sessions, see "Customizing the QuickTest Window Layout" on page 587.

---

## Removing a Movie from the Test Results

You can remove a stored movie from the results of a run. This reduces the size of the run results file. To remove a movie from the run results, choose **File** > **Remove Movie from Results**.

## Exporting Captured Movie Files

You can export a captured Screen Recorder movie to a file. The file is saved as an **.fbr** file. You can view **.fbr** files in the HP Micro Recorder (as described in "Viewing Screen Recorder Movie Files in the HP Micro Player" on page 644). You can also attach **.fbr** files to defects in Quality Center. Quality Center users who have the QuickTest Add-in for Quality Center installed can view the movies from Quality Center.

**To export a Screen Recorder movie:**

1 Choose **File > Export Movie to File**. The Save As dialog box opens, enabling you to change the default destination folder and rename the file, if required. By default, the file is named <**component name**> [<**name of run results**>], and is saved in the run results folder.

2 Click **Save** to save the exported (**.fbr**) file and close the dialog box.

### Viewing Screen Recorder Movie Files in the HP Micro Player

When you capture a movie of your run session using the Screen Recorder, the movie is saved as an **.fbr** file in your test results folder. You can export **.fbr** files to any location in your file system (as described in "Exporting Captured Movie Files" on page 643). You can also view these **.fbr** files without opening the QuickTest Test Results window, using the HP Micro Player.

**To play a Screen Recorder movie in the HP Micro Player:**

1 Perform one of the following:

- Double-click any **.fbr** file in Windows Explorer.
- Choose **Start > Programs > QuickTest Professional > Tools > HP Micro Player** and then choose **File > Open** in the Micro Player to select any **.fbr** file.

The movie opens in the HP Micro Player and begins playing.

2 Use the controls at the top of the window to access a particular location in the movie or to modify the volume settings.

## Finding Results Steps

The Find dialog box enables you to find specified steps such as errors or warnings from within the Test Results. You can select a combination of statuses to find, for example steps that are both **Passed** and **Done**.


The following options are available:

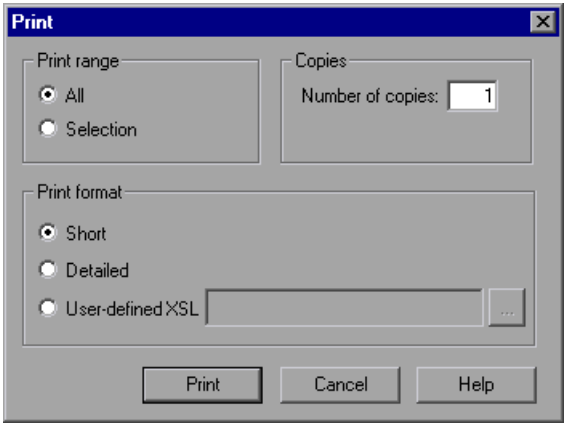
| Option | Description |
|---|---|
| **Failed** | Finds a failed step in the Test Results. |
| **Warning** | Find a step where a warning was issued. |
| **Passed** | Finds a passed step in the Test Results. |
| **Done** | Finds a step that has finished its run. |
| **Direction** | Indicates whether to search up or down within the steps of the Test Results. |

## Printing Run Session Results

You can print run results from the Test Results window. You can select the type of report you want to print, and you can also create and print a customized report.

**To print the run results:**

**1** Click the **Print** button or choose **File** > **Print**. The Print dialog box opens.




**2** Select a **Print range** option:

- **All.** Prints the results for the entire component.
- **Selection.** Prints the run results for the selected branch in the run results tree.

**3** Specify the **Number of copies** of the run results that you want to print.

**4** Select a **Print format** option:

- **Short.** Prints a summary line (when available) for each item in the run results tree. This option is only available if you selected **All** in step 2.
- **Detailed.** Prints all available information for each item in the run results tree, or for the selected branch, according to your selection in step 2.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the printed report, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 661.

---

**Note:** The **Print format** options are available only for run results created with QuickTest version 8.0 and later.

---

**5** Click **Print** to print the selected run results information to your default Windows printer.

## Previewing Test Results

You can preview run results on screen before you print them. You can select the type and quantity of information you want to view, and you can also display the information in a customized format.

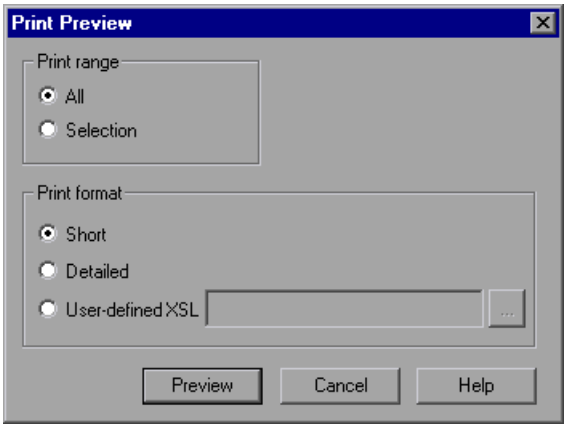
---

**Note:** The **Print Preview** option is available only for run results created with QuickTest version 8.0 and later.

---

**To preview the run results:**

**1** Choose **File** > **Print Preview**. The Print Preview dialog box opens.




**2** Select a **Print range** option:

- **All.** Previews the run results for the entire component.
- **Selection.** Previews run results information for the selected branch in the run results tree.

**3** Select a **Print format** option:

- **Short.** Previews a summary line (when available) for each item in the run results tree. This option is only available if you selected **All** in step 2.
- **Detailed.** Previews all available information for each item in the run results tree, or for the selected branch, according to your selection in step 2.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the preview, and the way it should appear. For more information, see “Customizing the Test Results Display” on page 661.

**4** Click **Preview** to preview the appearance of your run results on screen.

---

**Tip:** If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the **Page Setup** button in the Print Preview window and change the page orientation from **Portrait** to **Landscape**.

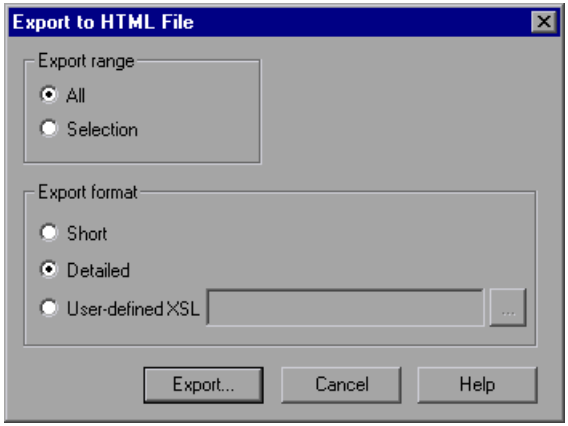
---

## Exporting Test Results

You can export the run results details to an HTML file. This enables you to view the run results even if the QuickTest environment is unavailable. For example, you can send the file containing the run results in an e-mail to a third-party who does not have QuickTest installed. You can select the type of report you want to export, and you can also create and export a customized report.

**To export the run results:**

**1** Choose **File** > **Export to HTML File**. The Export to HTML File dialog box opens.



**2** Select an **Export range** option:

- **All.** Exports the results for the entire component.
- **Selection.** Exports run result information for the selected branch in the run results tree.

**3** Select an **Export format** option:

- **Short.** Exports a summary line (when available) for each item in the run results tree. This option is only available if you selected **All** in step 2.
- **Detailed.** Exports all available information for each item in the run results tree, or for the selected branch, according to your selection in step 2.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the exported report, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 661.

---

**Note:** The **Export format** options are available only for run results created with QuickTest 8.0 and later.

---

**4** Click **Export**. The Save As dialog box opens, enabling you to change the default destination folder and rename the file, if required. By default, the file is named <name of component> [<name of run results>], and is saved in the run results folder.

**5** Click **Save** to save the HTML file and close the dialog box.

# Deleting Run Results

You can use the Test 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.

You can use this tool with a Windows-style user interface, or you can use the Windows command line to run the tool in the background (silently) to directly delete results that meet criteria that you specify.

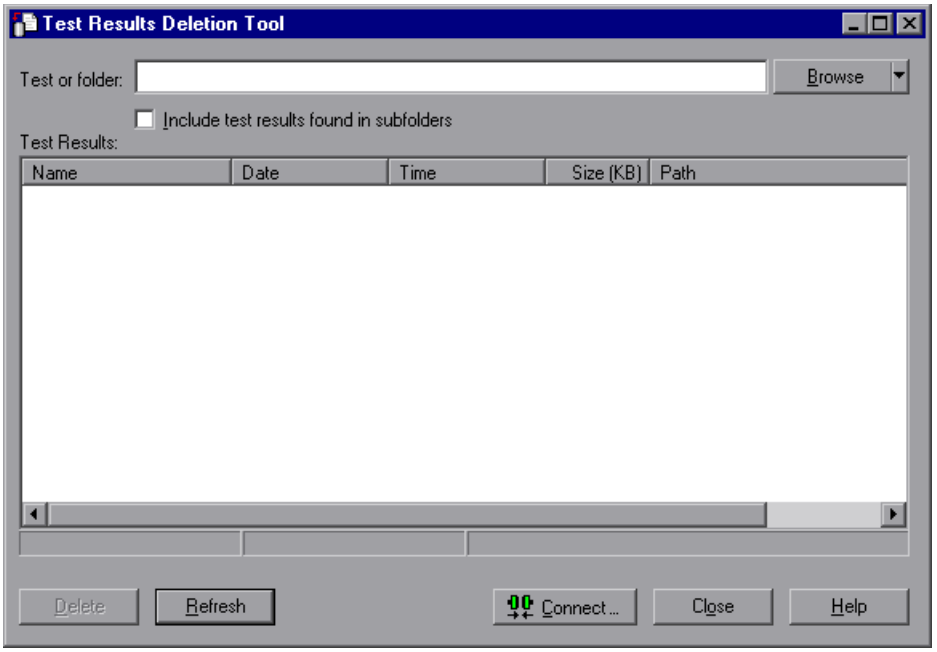
## Deleting Results Using the Test Results Deletion Tool

You can use the Test Results Deletion Tool to view a list of all the run session results in a specific location in your file system or in a Quality Center project. You can then delete any run results that you no longer require.

The Test 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 delete run results using the Test Results Deletion Tool:**

**1** Choose **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Test Results Deletion Tool** from the **Start** menu. The Tests Results Deletion Tool window opens.




**2** In the **Test or folder** box, specify the folder or specific test from which you want to delete test results. You can specify a full file system path or a full Quality Center path.

You can also browse to a test or folder as follows:

- To navigate to a specific test, click the **Browse** button or click the arrow to the right of the **Browse** button and select **Tests**.
- To navigate to a specific folder, click the arrow to the right of the **Browse** button and select **Folders**.

---

**Note:** To delete test results from a Quality Center database, click **Connect** to connect to Quality Center before browsing or entering the test path. Specify the Quality Center test path in the format [Quality Center] Subject\<folder name>\<test name>. For more information, see "Connecting to Your Quality Center Project" on page 44.

---

**3** Select **Include test results found in subfolders** if you want to view all tests results contained in subfolders of the specified folder.

---

**Note:** The **Include test results found in subfolders** check box is available only for folders in the file system. It is not supported when working with tests in Quality Center.

---

The test results in the specified test or folder are displayed in the Test Results box, together with descriptive information for each one. You can click a column's title in the Test Results box to sort test results based on the entries in that column. To reverse the order, click the column title again.

The Delete Test Results window status bar shows information regarding the displayed test results, including the number of results selected, the total number of results in the specified location and the size of the files.

**4** Select the test results you want to delete. You can select multiple test results for deletion using standard Windows selection techniques.

**5** Click **Delete**. The selected test results are deleted from the system and the Quality Center database.

---

**Tip:** You can click **Refresh** at any time to update the list of test results displayed in the Test Results box.

---

## Deleting Results Using the Windows Command Line

You can use the Windows command line to instruct the Test Results Deletion Tool to delete test results according to criteria you specify. For example, you may want to always delete test results older than a certain date or over a minimum file size.

**To run the Test Results Deletion Tool from the command line:**

Open a Windows command prompt and type <QuickTest installation path>\bin\TestResultsDeletionTool.exe, then type a space and type the command line options you want to use.

---

**Note:** If you use the -Silent command line option to run the Test Results Deletion Tool, all test results that meet the specified criteria are deleted. Otherwise, the Delete Test Results window opens.

---

### Command Line Options

You can use command line options to specify the criteria for the test results that you want to delete. Following is a description of each command line option.

---

**Note:** If you add command line options that contain spaces, you must specify the option within quotes, for example:
TestResultsDeletionTool.exe -Test "F:\Tests\Keep\web objects"

---

### -Domain *Quality_Center_domain_name*

Specifies the name of the Quality Center domain to which you want to connect. This option should be used in conjunction with the -Server, -Project, -User, and -Password options.

### -FromDate *results_creation_date*

Deletes test results created after the specified date. Results created on or before this date are not deleted. The format of the date is MM/DD/YYYY.

The following example deletes all results created after November 1, 2005.

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -FromDate "11/1/2005"
```

### -Log *log_file_path*

Creates a log file containing an entry for each test results file in the folder or component you specified. The log file indicates which results were deleted and the reasons why other results were not. For example, results may not be deleted if they are smaller than the minimum file size you specified.

You can specify a file path and name or use the default path and name. If you do not specify a file name, the default log file name is **TestResultsDeletionTool.log** in the folder where the Test Results Deletion Tool is located.

The following example creates a log file in **C:\temp\Log.txt**.

```
TestResultsDeletionTool.exe  -Silent -Log "C:\temp\Log.txt" -Test "C:\tests\test1"
```

The following example creates a log file named **TestResultsDeletionTool.log** in the folder where the Test Results Deletion Tool is located.

```
TestResultsDeletionTool.exe  -Silent -Log  -Test "C:\tests\test1"
```

### -MinSize *minimum_file_size*

Deletes test results larger than or equal to the specified minimum file size. Specify the size in bytes.

---

**Note:** The -MinSize option is available only for test results in the file system. It is not supported when working with components in Quality Center.

---

The following example deletes all results larger than or equal to 10000 bytes. Results that are smaller than 10000 bytes are not deleted.

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -MinSize "10000"
```

### -Name *result_file_name*

Specifies the names of the result files to be deleted. Only results with the specified names are deleted.

You can use regular expressions to specify criteria for the result files you want to delete. For more information on regular expressions and regular expression syntax, see the *HP QuickTest Professional User's Guide*.

The following example deletes results with the name **Res1**.

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -Name "Res1"
```

The following example deletes all results whose name starts with **Res** plus one additional character. (For example, **Res1** and **ResD** would be deleted. **ResDD** would not be deleted.)

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -Name "Res."
```

### -Password *Quality_Center_password*

Specifies the password for the Quality Center user name. This option should be used in conjunction with the -Domain, -Server, -Project, and -User options.

The following example connects to the **Default** Quality Center domain, using the server located at **http://QCServer/qcbin**, with the project named **Quality Center_Demo**, using the user name **Admin** and the password **PassAdmin**.

```
TestResultsDeletionTool.exe  -Domain "Default" -Server "http://QCServer/qcbin"
-Project "Quality Center_Demo" -User "Admin" -Password "PassAdmin"
```

### -Project *Quality_Center_project_name*

Specifies the name of the Quality Center project to which you want to connect. This option should be used in conjunction with the -Domain, -Server, -User, and -Password options.

### -Recursive

Deletes test results from all tests in a specified file system folder and its subfolders. When using the -Recursive option, the -Test option should contain the path of the folder that contains the tests results you want to delete (and not the path of a specific test).

The following example deletes all results in the **F:\Tests** folder and all of its subfolders.

```
TestResultsDeletionTool.exe  -Test "F:\Tests" -Recursive
```

---

**Note:** The -Recursive option is available only for folders in the file system. It is not supported when working with components stored in Quality Center.

---

### -Server *Quality_Center_server_path*

Specifies the full path of the Quality Center server to which you want to connect. This option should be used in conjunction with the -Domain, -Project, -User, and -Password options.

### -Silent

Instructs the Test Results Deletion Tool to run in the background (silently), without the user interface.

The following example instructs the Test Results Deletion Tool to run silently and delete all results located in **C:\tests\test1**.

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1"
```

### -Test *component_or_folder_path*

Sets the component or component path from which the Test Results Deletion Tool deletes test results. You can specify a component name and path, file system path, or full Quality Center path.

This option is available only when used in conjunction with the -Silent option.

---

**Note:** The -Domain, -Server, -Project, -User, and -Password options must be used to connect to Quality Center.

---

The following example opens the Test Results Deletion Tool with a list of the results in the **F:\Tests\Keep\webobjects** folder.

```
TestResultsDeletionTool.exe -Test "F:\Tests\Keep\webobjects"
```

The following example deletes all results in the Quality Center **Tests\webobjects** test:

```
TestResultsDeletionTool.exe -Domain "Default" -Server "http://QCServer/qcbin"
-Project "Quality Center_Demo592" -User "Admin" -Password "PassAdmin"
-Test "Subject\Tests\webobjects"
```

---

**Tip:** The -Test option can be combined with the -Recursive option to delete all test results in the specified file system folder and all its subfolders.

---

### -UntilDate *results_creation_date*

Deletes test results created before the specified date. Results created on or after this date are not deleted. The format of the date is MM/DD/YYYY.

This option is available only when used in conjunction with the -Silent option.

The following example deletes all results created before November 1, 2005.

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -UntilDate "11/1/2005"
```

### -User *Quality_Center_user_name*

Specifies the user name for the Quality Center project to which you want to connect. This option should be used in conjunction with the -Domain, -Server, -Project, and -Password options.

This option is available only when used in conjunction with the -Silent option.

# Manually Submitting Defects Detected During a Run Session to a Quality Center Project

When viewing the results of a run session, you can submit any defects detected to a Quality Center project directly from the Test Results window.

For more information on working with Quality Center and QuickTest, see the *HP QuickTest Professional User's Guide*. For more information on Quality Center, see the *HP Quality Center User's Guide*.

**To manually submit a defect to Quality Center:**

**1** Ensure that the Quality Center client is installed on your computer. (Enter the Quality Center Server URL in a browser and ensure that the Login screen is displayed.)

**2** Choose **Tools** > **Quality Center Connection** or click the **Quality Center Connection** button to connect to a Quality Center project. For more information on connecting to Quality Center, see "Connecting to Your Quality Center Project" on page 44.

---

**Note:** If you do not connect to a Quality Center project before proceeding to the next step, QuickTest prompts you to connect before continuing.

---

**3** Choose **Tools** > **Add Defect** or click the **Add Defect** button to open the Add Defect dialog box in the specified Quality Center project. The Add Defect dialog box opens.

**4** You can modify the defect information if required. Basic information on the component is included in the description:

| | |
|---|---|
| **Operating system :** | Windows 2000 |
| **Test path :** | [QualityCenter] Components\YE\ComponentWithDefect |

**5** Click **Submit** to add the defect information to the Quality Center project.

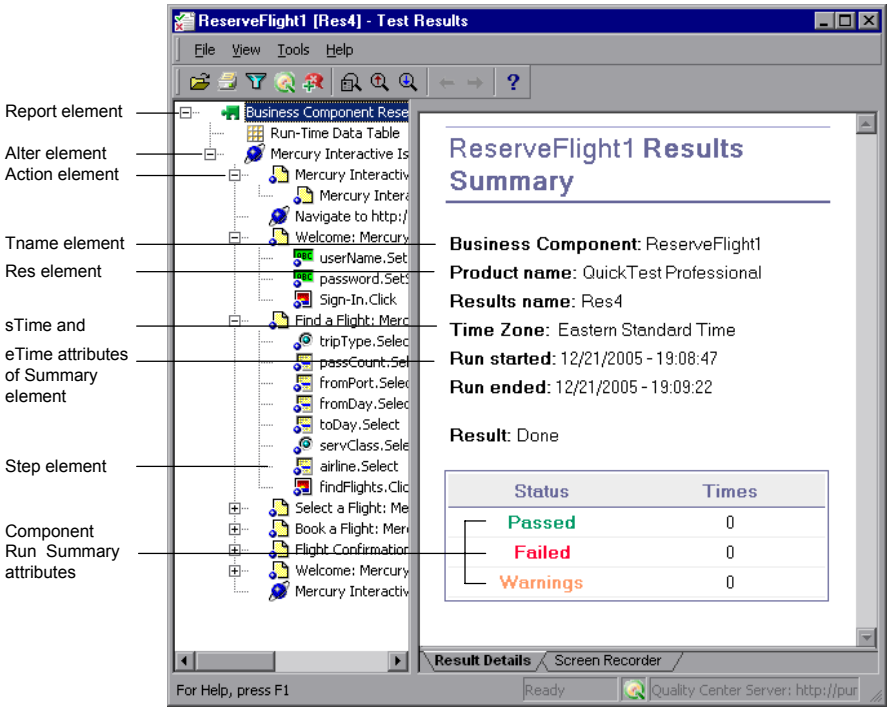
**6** Click **Close** to close the Add Defect dialog box.

# Customizing the Test Results Display

The results of each QuickTest run session are saved in a single **.xml** file (called **results.xml**). This **.xml** file stores information on each of the test result nodes in the display. The information in these nodes is used to dynamically create **.htm** files that are shown in the top-right pane of the Test Results window.

Each node in the run results tree is an element in the **results.xml** file. In addition, there are different elements that represent different types of information displayed in the test results. You can take test result information from the **.xml** file and use XSL to display the information you require in a customized format (either when printing from within the QuickTest Test Results window, when displaying test results in your own customized results viewer, or when exporting the test results to an HTML file).

The diagram below shows the correlation between some of the elements in the **.xml** file and the items they represent in the test results.




---

**Tip:** You can change the appearance (look and feel) of the Test Results window. For more information, see "Changing the Appearance of the Test Results Window" on page 632.

---

XSL provides you with the tools to describe exactly which test result information to display and exactly where and how to display, print or export it. You can also modify the **.css** file referenced by the **.xsl** file, to change the appearance of the report (for example, fonts, colors, and so forth).

For example, in the **results.xml** file, one element tag contains the name of an action, and another element tag contains information on the time at which the run session is performed. Using XSL, you could tell your customized test results viewer that the action name should be displayed in a specific place on the page and in a bold green font, and that the time information should not be displayed at all.

You may find it easier to modify the existing **.xsl** and **.css** files provided with QuickTest, instead of creating your own customized files from scratch. The files are located in <**QuickTest Installation Folder**>**dat**, and are named as follows:

- **PShort.xsl.** Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Short** option in the Print or Export to HTML File dialog boxes.
- **PDetails.xsl.** Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Detailed** option in the Print or Export to HTML File dialog boxes.
- **PSelection.xsl.** Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Selection** option in the Print or Export to HTML File dialog boxes.
- **PResults.css.** Specifies the appearance of the test results print preview. This file is referenced by all three **.xsl** files.

For more information on printing test results using a customized **.xsl** file, see "Printing Run Session Results" on page 646.

For more information on exporting the test results to an HTML file using a customized **.xsl** file, see "Exporting Test Results" on page 649.

For information on the structure of the XML schema, and a description of the elements and attributes you can use to customize the test results reports, see the XML Report Help (**Help** > **QuickTest Professional Help** > **QuickTest Advanced References** > **QuickTest Test Results Schema**).

# 24

# Analyzing Run Session Results

You can analyze the results of a run session using the report of major events that occurred during the run session.

---

**Note:** You cannot view business process test run results when you open the Test Results window from QuickTest. To view run results for a business process test, select the results for the iteration you want to view and open them from within Quality Center.

---

**This chapter includes:**

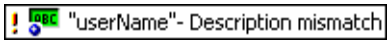
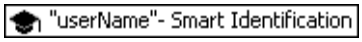
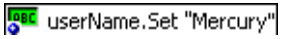


## Analyzing Smart Identification Information in the Test Results

If the recorded description does not enable QuickTest to identify the specified object in a step, and a Smart Identification definition is defined (and enabled) for the object, then QuickTest tries to identify the object using the Smart Identification mechanism. The following examples illustrate two possible scenarios.

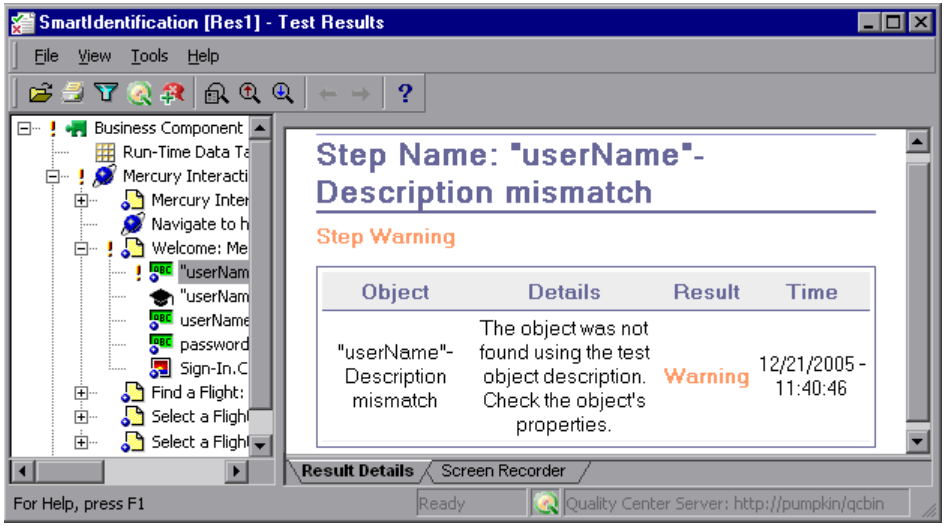
### Smart Identification—No Object Matches the Recorded Description

If QuickTest successfully uses Smart Identification to find an object after no object matches the recorded description, the Test Results receive a warning status and include the following information:

| In the results tree: | In the results details: |
|---|---|
| A description mismatch icon for the missing object. For example:<br>  | An indication that the object (for example, the userName WebEdit object) was not found. |
| A Smart Identification icon for the missing object. For example:<br>  | An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to modify the recorded test object description, so that QuickTest can find the object using the description in future run sessions. |
| The actual step performed. For example:<br>  | Normal result details for the performed step. |

For more information on the Smart Identification mechanism, see Chapter 5, “Configuring Object Identification.”

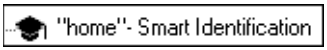
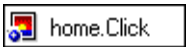
The image below shows the results for a component in which Smart Identification was used to identify the userName WebEdit object after one of the recorded description property values changed.



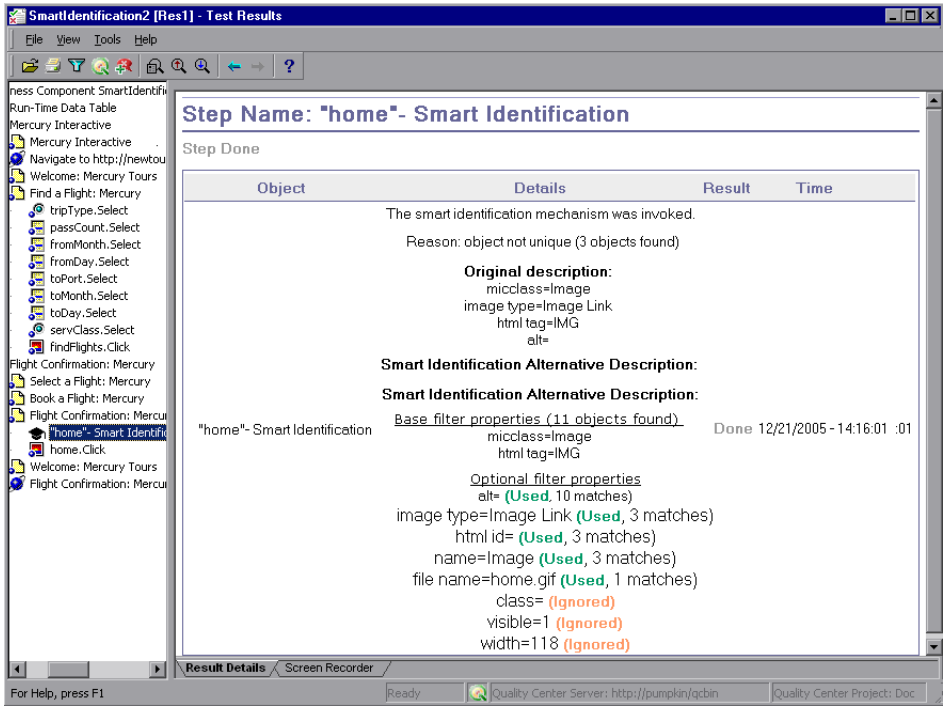

## Smart Identification—Multiple Objects Match the Recorded Description

If QuickTest successfully uses Smart Identification to find an object after multiple objects are found that match the recorded description, QuickTest shows the Smart Identification information in the Test Results window. The step still receives a passed status, because in most cases, if Smart Identification was not used, the test object description plus the ordinal identifier could have potentially identified the object.

In such a situation, the Test Results show the following information:

| In the results tree: | In the results details: |
|---|---|
| A Smart Identification icon for the missing object. For example:<br>  | An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to create a unique object description for the object, so that QuickTest can find the object using the description in future run sessions. |
| The actual step performed. For example:<br>  | Normal result details for the performed step. |

The image below shows the results for a component in which Smart Identification was used to uniquely identify the Home object after the recorded description resulted in multiple matches.




If the Smart Identification mechanism cannot successfully identify the object, the component fails and a normal failed step is displayed in the Test Results.

# Viewing Checkpoint Results

By adding checkpoints to your component, you can compare expected values in, for example, Web pages, text strings, and object properties to the values of these elements in your application. This enables you to ensure that your application functions as desired.

When you run the component, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails, which causes the component to fail. You can view the results of the checkpoint in the Test Results window.

**To view the results of a checkpoint:**

**1** Display the test results for your component in the Test Results window. For more information, see "Viewing the Results of a Run Session" on page 633.

**2** In the left pane of the Test Results window, expand the branches of the run results tree and click the branch for the checkpoint whose results you want to view. The checkpoint results are displayed in the Test Results window.

---

**Note:** By default, the bottom right part of the Test Results window displays information on the selected checkpoint only if it has the status **Failed**. You can change the conditions for when a step's image is saved, in the Run tab of the Options dialog box. For more information, see "Setting Run Testing Options" on page 591.

---

The information in the Test Results window and the available options are determined by the type of checkpoint you selected. For more information, see:

- "Analyzing Standard Checkpoint Results" on page 671
- "Analyzing Bitmap Checkpoint Results" on page 672

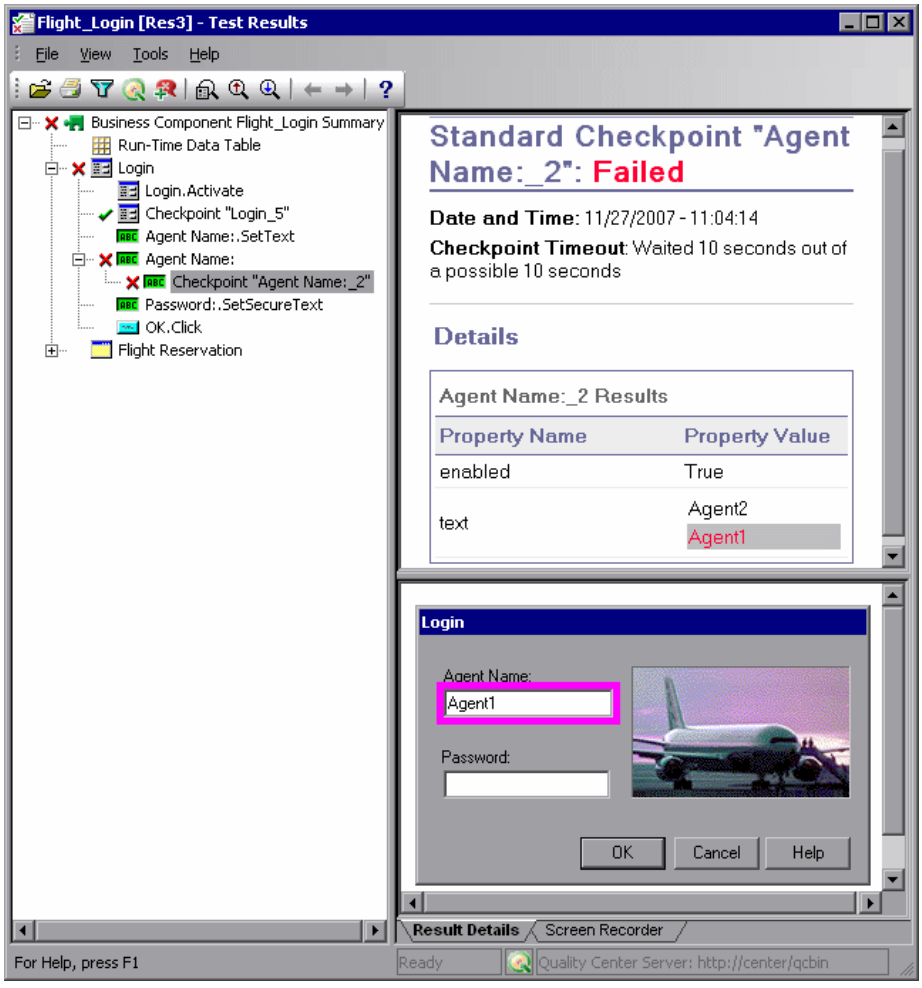
**3** Choose **File** > **Exit** to close the Test Results window.

For more information on checkpoints, see Chapter 16, "Understanding Checkpoints."

## Analyzing Standard Checkpoint Results

By adding standard checkpoints to your components, you can compare the expected values of object properties to the object's current values during a run session. If the results do not match, the checkpoint fails. For more information on standard checkpoints, see "Checking Object Property Values" on page 557.

You can view detailed results of the standard checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see "Viewing Checkpoint Results" on page 670.



The top right pane displays detailed results of the selected checkpoint, including its status (**Passed** or **Failed)**, the date and time the checkpoint was run, and the portion of the checkpoint timeout interval that was used (if any). It also displays the values of the object properties that are checked, and any differences between the expected and actual property values.

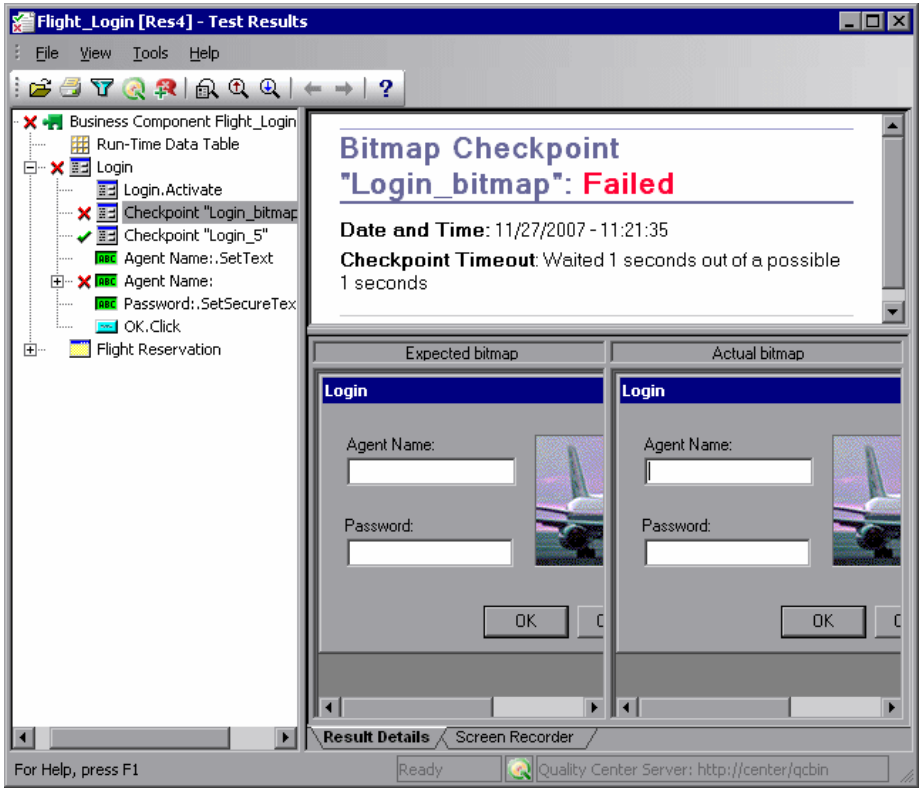
The bottom right pane displays the image capture for the checkpoint step (if available).

In the above example, the details of the failed checkpoint indicate that the expected results and the current results do not match. The expected value of the agent name is **Agent2**, but the actual value is **Agent1**.

## Analyzing Bitmap Checkpoint Results

By adding bitmap checkpoints to your components, you can check the appearance of elements in your application by matching captured bitmaps. When you run your component, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails. For more information on bitmap checkpoints, see Chapter 18, “Checking Bitmaps.”

You can view detailed results of the bitmap checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see "Viewing Checkpoint Results" on page 670.




The top right pane displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any).

The bottom right pane shows the expected and actual bitmaps that were compared during the run session.

# Viewing Parameterized Values and Output Value Results in the Test Results Window

You can view information on parameterized values and the results of output value steps in the Test Results window.

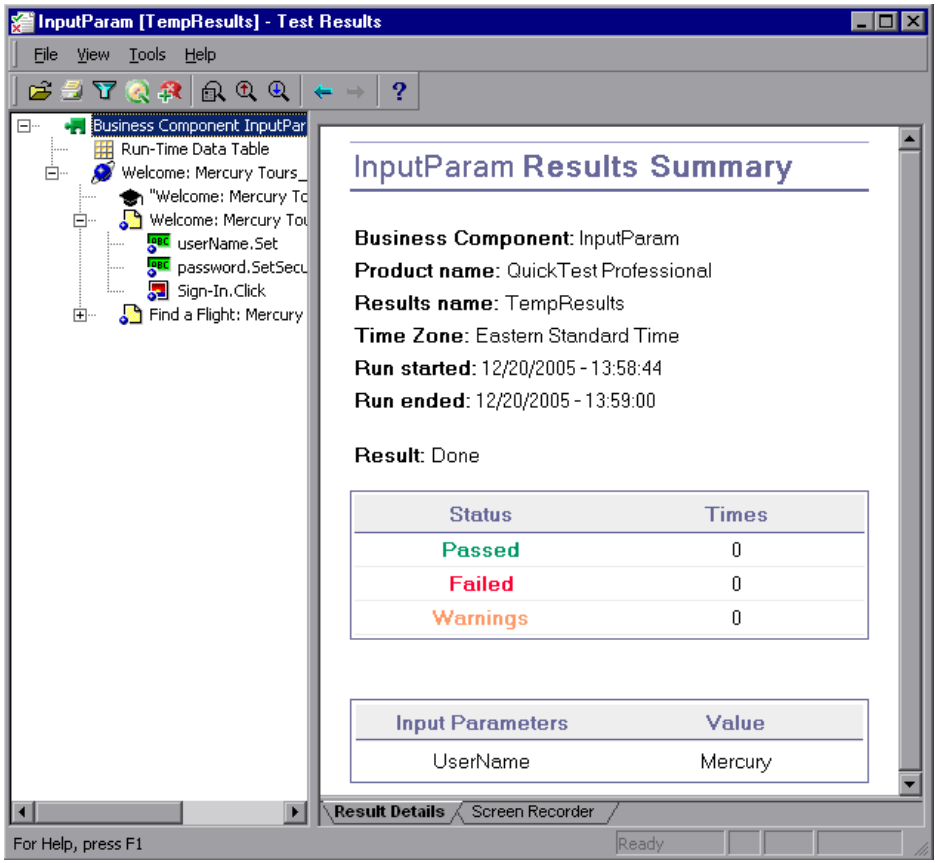
## Viewing Parameterized Values in the Test Results Window

A **parameter** is a variable that is assigned a value from within a component. You can view the values for the parameters defined in your component in the Test Results window.

**To view parameterized values:**

1. Display the run results for your component in the Test Results window. For more information, see "Viewing the Results of a Run Session" on page 633.
2. In the left pane of the Test Results window, select the root node, which contains the name of the component.

The name and value of the input parameters are displayed at the bottom of the right pane.




The example above shows the input parameter UserName defined for the component with the value Mercury.

For more information on defining and using parameters in your components, see “Working with Parameters” on page 533.

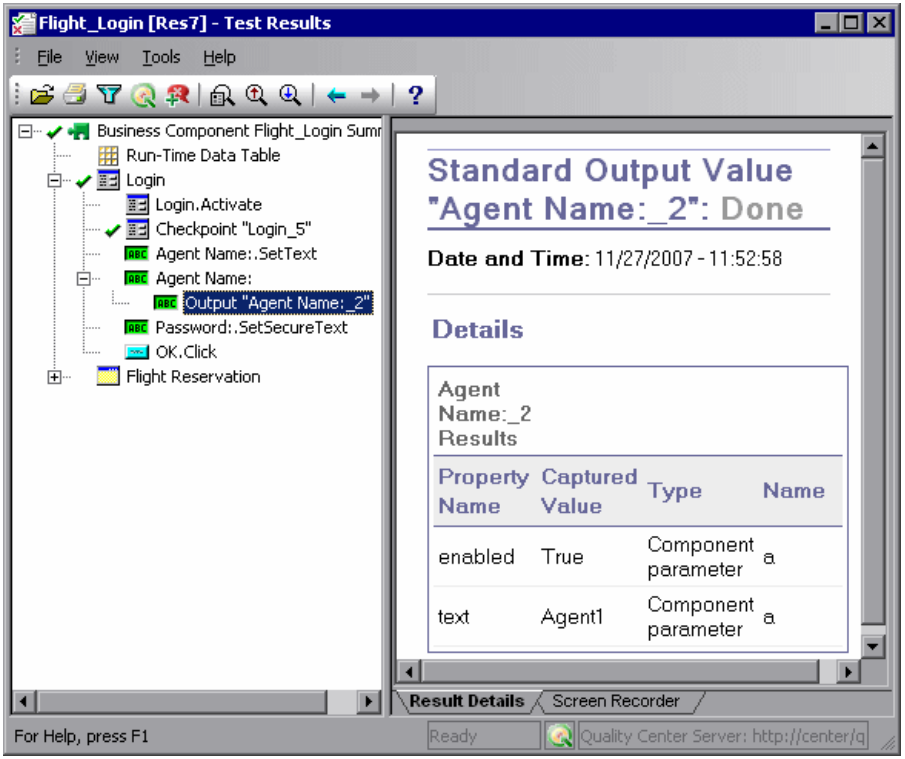
## Viewing Output Value Results in the Test Results Window

An **output value** is a step in which one or more values are captured during the run session for use at another point in the run. When one of the values is needed later in the run as input, QuickTest retrieves it from the specified output location.

**To view the results of an output value step:**

**1** Display the run results for your component in the Test Results window. For more information, see "Viewing the Results of a Run Session" on page 633.

**2** In the left pane of the Test Results window, expand the branches of the run results tree and click the branch for the output value step whose results you want to view. The output value results are displayed in the Test Results window.



The right pane displays detailed results of the selected output value step, including its status, and the date and time the output value step was run. It also displays the details of the output value, including the value that was captured during the run session, its type, and its name.

For more information on output values, see Chapter 19, “Outputting Values.”

# Part VII

## Maintaining and Debugging Components

# 25

# Debugging Components and Function Libraries

By controlling and debugging your run sessions, you can identify and handle defects in your components, function libraries, and registered user functions.

---

**Note:** Before you can debug components in QuickTest, you must enable integration between QuickTest and your Quality Center project by selecting the **Allow other HP products to run tests and components** check box (from QuickTest, choose **Tools** > **Options** > **Run**).

---

**This chapter includes:**

# About Debugging Components and Function Libraries

After you create a component or function library (including registered user functions), you should check that they run smoothly, without errors in syntax or logic. To debug a function library, you must first associate it with a component via its application area and then debug it from that component.

To detect and isolate defects in a component or function library, you can control the run session using the **Pause** command as well as various step commands that enable you to step into, over, and out of a specific step.

You can use the **Debug from Step** command to begin your debug session at a specific point in your component. You can also use the **Run to Step** command to pause the run at a specific point in your component. You can set breakpoints, and then enable and disable them as you debug different parts of your component or function library.

When the component or function library run stops at a breakpoint, you can use the Debug Viewer to check and modify the values of VBScript objects and variables. Also, if QuickTest displays a run error message during a run session, you can click the **Debug** button on the error message to suspend the run and debug the component or function library.

You can also use the **Run from Step** command to run your component or function library from a selected step to the end. This enables you to check a specific section of your application or to confirm that a certain part of your component or function library runs smoothly. For more information, see "Running Part of Your Component" in the *HP QuickTest Professional User's Guide*.

---

**Tip:** You can use the Screen Recorder to capture a movie of your application as it is being tested. For more information, see "Capturing and Viewing Still Images and Movies of Your Application" on page 640.

---

**Notes:**

- While the component and function libraries are running in debug mode, they are read-only. You can modify the content after you stop the debug session (not when you pause it). If needed, you can enable the function library for editing (**File > Enable Editing**) after you stop the session. For more information, see "Editing a Read-Only Function Library" on page 383. After you implement your changes, you can continue debugging your component and function libraries.
- If you perform a file operation (for example, open a different component or create a new component), the debug session is stopped.
- In QuickTest, when you open a component, QuickTest creates a local copy of the external resources that are saved to your Quality Center project. Therefore, any changes you apply to any external resource that is saved in your Quality Center project, such as a function library, will not be implemented in the component until the component is closed and reopened. (An external resource is any resource that was not created using QuickTest, such as, a function library created in an external editor.)

## Slowing a Debug Session

During a run session, QuickTest normally runs steps quickly. While you are debugging a component or function library, you may want QuickTest to run the steps more slowly so you can pause the run when needed or perform another task. You can specify the time (in milliseconds) QuickTest pauses between each step by modifying the **Delay each step execution by** option in the Run tab of the Options dialog box (**Tools > Options**). For more information on the Run tab options, see "Setting Run Testing Options" on page 591.

# Using the Single Step Commands

You can run a single step of a component or function library using the **Step Into**, **Step Out**, and **Step Over** commands.

---

**Tip:** To display the Debug toolbar, choose **View** > **Toolbars** > **Debug**.

---

### Step Into

Choose **Debug** > **Step Into**, click the **Step Into** button, or press F11 to run only the current line of the active component or function library. If the current line of the active component or function library calls a function, the called function is displayed in the QuickTest window, and the function library pauses at the first line of the called function.

### Step Out

Choose **Debug** > **Step Out**, click the **Step Out** button, or press SHIFT+F11 only after using **Step Into** to enter a user-defined function. **Step Out** runs to the end of the user-defined function, then returns to the calling component or function library and pauses the run session.

### Step Over

Choose **Debug** > **Step Over**, click the **Step Over** button, or press F10 to run only the current step in the active component or function library. When the current step calls a user-defined function, the called function is executed in its entirety, but the called function script is not displayed in the QuickTest window.

## Using the Single Step Commands - An Example

Follow the instructions below to create a sample function library and run it using the **Step Into, Step Out, and Step Over** commands.

**To create the sample function library:**

1 Choose **File** > **New** > **Application Area**. A new application area opens. (For more information, see "Creating an Application Area" on page 417.)

2 Create a new function library named **SampleFL.qfl** and save it to your Quality Center project. (For more information, see "Managing Function Libraries" on page 426.)

3 Open **SampleFL.qfl** and enter the following lines exactly:

```
public Function myfunc()
msgbox "one"
msgbox "two"
msgbox "three"
End Function
```

4 Associate the function library with the component's application area by choosing **File** > **Associate Library** '<**Function Library Name**>' **with** '<**Application Area Name**>', or right-clicking and choosing **Associate Library** '<**Function Library Name**>' **with** '<**Application Area Name**>'. QuickTest associates the function library with your application area.

**To run the component using the Step Into, Step Out, and Step Over commands:**

1 Create a new component based on the application area you created in the previous section.

2 Insert three identical steps. For each step:

- In the **Item** cell, select **Operation**.
- In the **Operation** cell, select **myfunc**.

3 Open the **SampleFL.qfl** function library, if it is not already open, or click the tab for the **SampleFL.qfl** function library to bring it into focus.

4. Add a breakpoint on the first line of the component (the first call to the myfunc function) by pressing F9 (**Insert/Remove Breakpoint**). The breakpoint symbol is displayed in the left margin. For more information, see "Setting Breakpoints" on page 691.
5. Run the component. The component pauses at the breakpoint.
6. Press F11 (**Step Into**). The execution arrow points to the first line within the function (msgbox "one").
7. Press F11 (**Step Into**) again. A message box displays the text one.
8. Click **OK** to close the message box. The execution arrow moves to the next line in the function.
9. Continue pressing F11 (**Step Into**) until the execution arrow leaves the function and is pointing to the eighth line in the script (the second call to the myfunc function).
10. Press F11 (**Step Into**) to enter the function again. The execution arrow points to the first msgbox line within the function.
11. Press SHIFT+F11 (**Step Out**). Three message boxes open. 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 last line in the test.
12. Press F10 (**Step Over**). The three message boxes open again. The execution arrow remains on the last line in the test.

# Using the Run to Step and Debug from Step Commands

In addition to stepping into, out of, and over a step while debugging, you can use the **Run to Step** and **Debug from Step** commands to instruct QuickTest to run a component (including any associated function library) until it reaches a particular step, or to begin debugging from a specific step.

## Run to Step

You can instruct QuickTest to run from the beginning of the component—or from the current location in the component—and to stop at a particular step. This is similar to adding a temporary breakpoint to a step. For example, if you are running a component and any associated function library in debug mode, one step at a time, you may want to run four consecutive steps and then stop at the fifth step.

You can use this option while editing or debugging your component.

**To instruct QuickTest to run to a particular step:**

➤ Insert your cursor in the step in which you want QuickTest to stop the run and choose **Debug** > **Run to Step** or press CTRL+F10, or

➤ Right-click in the step in which you want QuickTest to stop the run and choose **Run to Step** from the context menu.

---

**Note:** If while editing your component, you use the **Run to Step** option, the Run dialog box opens, enabling you to specify the results location and the input parameter values for the debug run session. For more information, see step2 in the "Debug from Step" section, below.

---

## Debug from Step

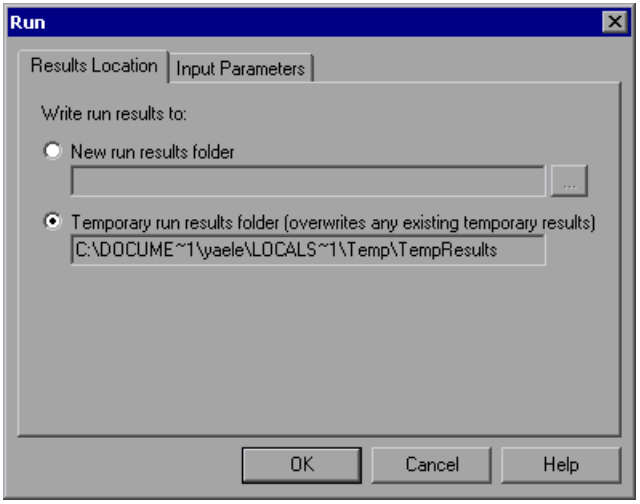
You can instruct QuickTest to begin your debug session from a particular step instead of beginning the run at the start of the component. Before you start debugging from a specific step, make sure that the application is open to the location from which you want to begin debugging. You can begin debugging from a specific step in your component when editing a component.

**To instruct QuickTest to run from a particular step:**

**1** Select the step from which you want to begin debugging:

- ➤ Insert your cursor in the step where you want QuickTest to start the run and choose **Debug** > **Debug from Step**, or
- ➤ Right-click in the step where you want QuickTest to start the run and choose **Debug from Step** from the context menu.

The Run dialog box opens.




**2** If applicable, specify the results location and the input parameter values for the debug run session. By default, the **Temporary run results folder** option is selected.

For more information on the tabs in the Run dialog box, see "Understanding the Results Location Tab" on page 620, and "Understanding the Input Parameters Tab" on page 621.

**3** Click **OK**. The Run dialog box closes and the debug run session starts. You can use any of the QuickTest debugging options, such as **Step Into**, **Step Over**, and **Run to Step**.

By default, when the run session ends, the Test Results window opens. For more information on viewing the run results, see Chapter 23, "Viewing Run Session Results."

---

**Note:** If you cleared the **View results when run session ends** check box in the Run tab of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 20, "Setting Global Testing Options."

---

## Pausing a Run Session

You can temporarily suspend a run session by choosing **Debug** > **Pause** or clicking the **Pause** button. A paused component or function library stops running when all previously interpreted steps have been run.

To resume running a paused run, click the **Run** button, choose **Automation** > **Run**, or press **F5**. The run continues from the point it was suspended.

---

**Tip:** You can also stop a run session by clicking the **Stop** button or choosing **Automation** > **Stop**. After the run session stops, the Test Results window opens (unless you selected not to view results at the end of a run session (**Tools** > **Options** > **Run** tab)).

---

# Using Breakpoints

You can use breakpoints to instruct QuickTest to pause a run session at a predetermined place in a component or function library. QuickTest pauses the run when it reaches the breakpoint, before executing the step. You can then examine the effects of the run up to the breakpoint, make any necessary changes, and continue running the component or function library from the breakpoint.

You can use breakpoints to:

- suspend a run session and inspect the state of your application
- mark a point from which to begin stepping through a component or function library using the step commands

You can set breakpoints, and you can temporarily enable and disable them. After you finish using them, you can remove them from your component or function library.

---

**Note:** Breakpoints are applicable only to the current QuickTest session and are not saved with your component or function library.

---

## Setting Breakpoints

By setting a breakpoint, you can pause a run session at a predetermined place in a component or function library. A breakpoint is indicated by a filled red circle icon in the left margin adjacent to the selected step.

**To set a breakpoint:**

Perform one of the following:

- Click in the left margin of a step in the component or function library where you want the run to stop, or
- Click a step and then:

  - Click the **Insert/Remove Breakpoint** button
  - Choose **Debug** > **Insert/Remove Breakpoint**

The breakpoint symbol is displayed in the left margin of the component or function library.

---

**Tip:** You can also use the **Enable/Disable Breakpoint** option to add a breakpoint to a step. For more information, see “Enabling and Disabling Breakpoints” on page 692.

---

## Enabling and Disabling Breakpoints

You can instruct QuickTest to ignore an existing breakpoint during a debug session by temporarily disabling the breakpoint. Then, when you run your component or function library, QuickTest runs the step containing the breakpoint, instead of stopping at it. When you enable the breakpoint again, QuickTest pauses there during the next run. This is particularly useful if your component or function library contains many steps, and you want to debug a specific part of it.

You can enable or disable breakpoints individually or all at once. For example, suppose you add breakpoints to various steps throughout your component or function library, but for now you want to debug only a specific part of your document. You could disable all breakpoints in your component or function library, and then enable breakpoints only for specific steps. After you finish debugging that section of your document, you could disable the enabled breakpoints, and then enable the next set of breakpoints (in the section you want to debug). Because the breakpoints are disabled and not removed, you can find and enable any breakpoint, as needed.

An enabled breakpoint is indicated by a filled red circle icon in the left margin adjacent to the selected step.

A disabled breakpoint is indicated by an empty circle icon in the left margin adjacent to the selected step.

**To enable/disable a specific breakpoint:**

1. Click in the line containing the breakpoint you want to disable/enable.
2. Choose **Debug** > **Enable/Disable Breakpoint** or press CTRL+F9. The breakpoint is either disabled or enabled (depending on its previous state).

**To enable/disable all breakpoints:**

Choose **Debug** > **Enable/Disable All Breakpoints** or click the **Enable/Disable All Breakpoints** button. If at least one breakpoint is enabled, QuickTest disables all breakpoints in the component or function library. Alternatively, if all breakpoints are disabled, QuickTest enables them.

## Removing Breakpoints

You can remove a single breakpoint or all breakpoints defined for the current component or function library.

**To remove a single breakpoint:**

Perform one of the following:

- Click the breakpoint.
- Click the line in your component or function library with the breakpoint symbol and:

  - Click the **Insert/Remove Breakpoint** button.
  - Choose **Debug** > **Insert/Remove Breakpoint**.

The breakpoint symbol is removed from the left margin of the QuickTest window.

**To remove all breakpoints:**

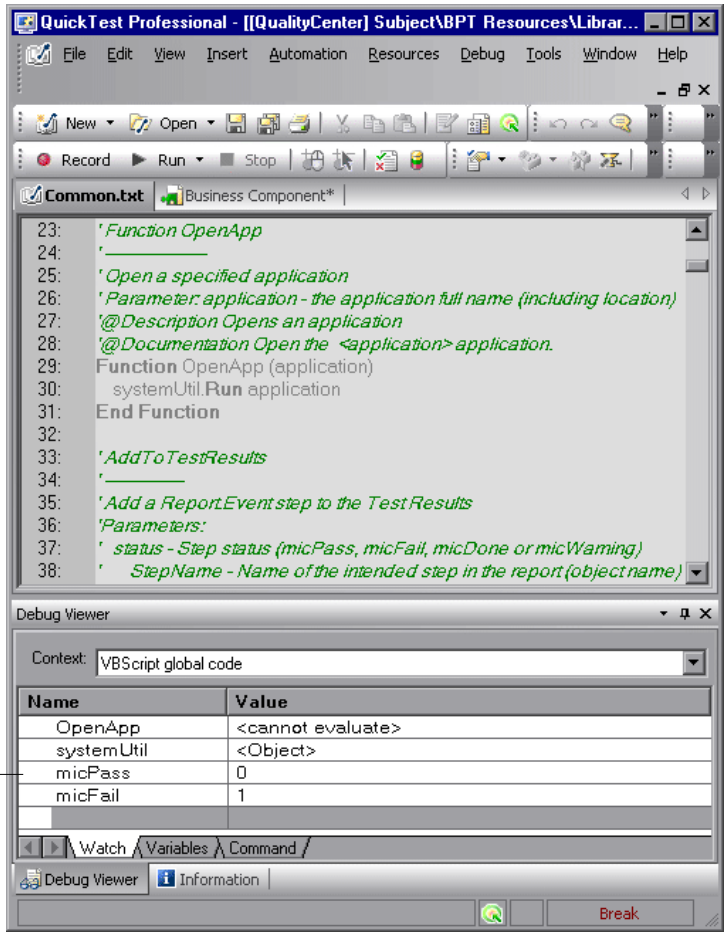
Click the **Clear All Breakpoints** button, or choose **Debug** > **Clear All Breakpoints**. All breakpoint symbols are removed from the left margin of the QuickTest window.

# Using the Debug Viewer

You use the Debug Viewer pane to view, set, or modify the current value of objects or variables in your function library, when it stops at a breakpoint, or when a step fails and you select the **Debug** option. The Debug Viewer is useful for debugging operations (functions) in a business component, but is not intended for use with other types of component steps.

**To open the Debug Viewer pane:**

Choose **View** > **Debug Viewer** or click the **Debug Viewer** button. The Debug Viewer pane opens.



The Debug Viewer tabs are used to display the values of variables and objects in the main script of the selected subroutine.

## Watch Tab

You can view the current value of any variable or VBScript object in your function library by adding it to the Watch tab. As you continue stepping into the subsequent steps in your function library, QuickTest automatically updates the Watch tab with the current value for any object or variable whose value changes. You can also change the value of the variable manually when the function library pauses at a breakpoint.

**To add an expression to the Watch tab:**

Perform one of the following:

- Click the expression and choose **Debug** > **Add to Watch**.
- Click the expression and press CTRL+T.
- Right-click the expression and choose **Add to Watch** from the context menu.
- In the Watch tab, paste or type the name of the object or variable into the **Name** column and press ENTER to view the current value in the **Value** column.

---

**Note:** You can add an expression to the Watch tab from a function library (and not from a business component).

---

### Variables Tab

QuickTest automatically displays the current value of all variables in the current function in the Variables tab—up to the point where the function library is stopped or paused. For example, if you are stepping through a function, as you step into each step, QuickTest adds the current value for any step variable to the Variables tab grid. As you continue stepping into the subsequent steps, QuickTest automatically updates the value displayed in the Variables tab for any variable whose value changes. You can also change the value of the variable manually, during the breakpoint pause.

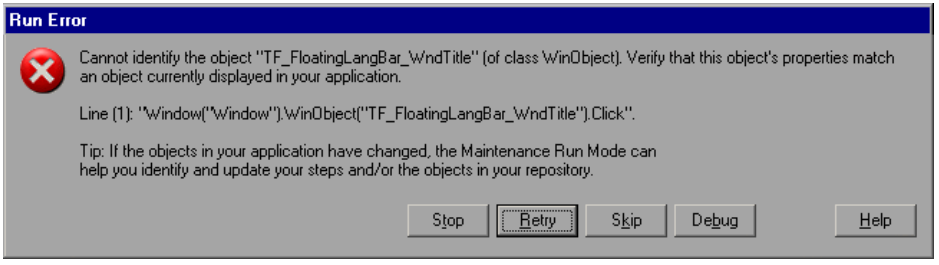
### Command Tab

Use the Command tab to execute a line of script to set or modify the current value of a variable or VBScript object in your function library. When the run continues, QuickTest uses the value that you set.

## Handling Run Errors

There are two types of Run Error message boxes that can be displayed during a run session. One is displayed if the problem is a pure VBScript syntax error. When a syntax run error message box is displayed, click OK in the message box and address the error in your step.

The other message box can be displayed in a number of situations, and offers information about the error and a number of buttons for dealing with errors encountered:



- **Stop.** Stops the run session. The run results are displayed if QuickTest is configured to show run results after the run.
- **Retry.** QuickTest attempts to perform the step again. If the step succeeds, the run continues.
- **Skip.** QuickTest skips the step that caused the error, and continues the run from the next step.
- **Debug.** QuickTest suspends the run, enabling you to debug the component and any associated function library that contains a function called by the component.

  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 component or function library stopped, or you can use the step commands to control the remainder of the run session.
- **Help.** Opens the QuickTest troubleshooting Help for the displayed error message. After you review the Help topic, you can select another button in the error message box.

  The message box also recommends that you consider using Maintenance Mode if you think the error is due to intentional changes in your application and requires that you update multiple steps in your component or objects in your repository. For more information, see "Running Components with the Maintenance Run Wizard" on page 704.

# Practicing Debugging a Function

Suppose you create a function that defines variables that will be used in other parts of your function library. You can add breakpoints to the function to see how the value of the variables changes as the function library runs. To see how the function library handles the new value, you can also change the value of one of the variables during a breakpoint.

### Step 1: Create a New Function

Open a new function library and create a new function called **SetVariables**. For more information on working with functions, see Chapter 11, "Working with User-Defined Functions and Function Libraries."

Enter the VBScript code, as follows:

```
Function SetVariables

Dim a
a="hello"
b="me"
MsgBox a

EndFunction
```

### Step 2: Associate the Function Library with an Application Area

1. Make sure the function library is in focus. (If it is not in focus, activate it by clicking the function library's tab or choosing it from the **Window** menu.)
2. Choose **File** > **Associate Library** '**<Function Library Name>**' **with** '**<Application Area Name>**', or right-click and choose **Associate Library** '**<Function Library Name>**' **with** '**<Application Area Name>**'. QuickTest associates the function library with your application area.

### Step 3: Add a Call to the Function in the Component

Add a call to the function by inserting a new operation and choosing **SetVariables** from the **Operation** list.

### Step 4: Add Breakpoints

Add breakpoints at the lines containing the text b="me" and MsgBox a. For more information on adding breakpoints, see "Setting Breakpoints" on page 691.

### Step 5: Begin Running the Component

Run the component. The component or function library stops at the first breakpoint, before executing that step (line of script).

### Step 6: Check the Value of the Variables in the Debug Viewer Pane

1. Choose **View** > **Debug Viewer** to open the Debug Viewer pane, if it is not already open. Then select the **Watch** tab on the Debug Viewer pane.
2. In the document pane, select the variable **a** and choose **Debug** > **Add to Watch**. QuickTest adds the variable a to the Watch tab. The **Value** column indicates that the value of **a** is currently **hello**, because the breakpoint stopped after the value of variable **a** was initiated.
3. In the document pane, select the variable **b** and choose **Debug** > **Add to Watch**. QuickTest adds the variable b to the Watch tab. The **Value** column indicates **Variable is undefined:** '**b**', because the component stopped before variable **b** was declared.
4. Select the **Variables** tab in the Debug Viewer pane. Both **SetVariables** (with the value **Empty**) and variable **a** (with the value **hello**) are displayed. Variable **b** is not displayed because the component stopped before variable **b** was declared.

### Step 7: Check the Value of the Variables at the Next Breakpoint

Click the **Run** button to continue running the component. The component stops at the next breakpoint. Note that the values of variables **a** and **b** have both been updated in the Watch and Variables tabs.

### Step 8: Modify the Value of a Variable Using the Command Tab

Select the **Command** tab in the Debug Viewer pane.

Type: a="This is the new value of a" at the command prompt, and press ENTER on the keyboard. Click the **Run** button to continue running the component. The message box that appears displays the new value of **a**.

# 26

# Maintaining Components

QuickTest provides tools that enable you to maintain your components as the application you are testing changes. For example, your application's objects may change their properties or descriptions, or they may no longer exist. The expected values of your component's checkpoints may also need updating based on changes in your application. This chapter describes how you can use QuickTest's tools to update and maintain your components.

**This chapter includes:**



## Why Components Fail

Components fail when QuickTest encounters conditions in a component it did not expect. In many cases this is due to the application being tested not functioning properly. QuickTest then provides you with test results that assist you in understanding how to fix your application.

Sometimes a component fails because the application being tested has changed from when the component was created and the QuickTest component needs to be updated to reflect those changes. QuickTest provides tools that help identify and resolve some of these issues.

## Object Changes

When QuickTest runs a step in a component, it looks for the object referred to by that step, in the object repositories associated with that component. Using the description of the object in the repository, QuickTest attempts to identify that object in the application.

QuickTest may not be able to identify the object in the application for a number of reasons.

### The Object Does Not Exist in the Application

QuickTest cannot find an object in the application that matches the description of the object in the object repository. Maintenance Mode enables you to identify the object that you want your component to use.

### The Parent Object Changed

QuickTest cannot find an object in the application that matches and has the same hierarchy as the object in the object repository. Maintenance Mode enables you to identify the object that you want your component to use.

### The Object Description Property Values Changed

QuickTest cannot find an object in the application that is similar to, and has the same description property values as the object in the object repository. Maintenance Mode enables you to identify the object that you want your component to use.

### The Object Does Not Exist in the Object Repository

QuickTest looks for the object to which the component refers, in the associated object repositories before attempting to identify that object in the application. If the object cannot be found in any associated object repository, QuickTest raises the Run Error dialog box, informing you that the object does not exist in the object repository. The missing object needs to be added manually to the object repository or you need to change your step.

### The Description Set of the Object Needs to Change

QuickTest uses a set of properties to identify objects in the application. If the set of identification properties for the object in the object repository does not provide a unique description matching an object in the application, QuickTest will be unable to find the object. Update Run Mode enables you to update the set of identification properties for the objects in your component to match those defined in the Object Repository dialog box.

## Checkpoint Changes

Checkpoints fail when they encounter conditions in the application being tested that are unexpected. In many cases this is due to the application not functioning properly. QuickTest provides you with test results that assist you in understanding how to fix your application.

Sometimes checkpoints fail because the application has changed since the component was created and the QuickTest checkpoints need to be updated to reflect those changes. Update Run Mode enables you to update the checkpoints in your component to reflect changes in the application.

For example, suppose your application has an edit box whose label is Name and whose value should be Michael. You can create one checkpoint to check if the edit box's label is Name, and another checkpoint to check if the edit box has the value Michael. If the checkpoint that checks the value of the edit box fails, (it contains Suzy) the application is not functioning properly and you can use QuickTest's test results to determine how to fix the application. If the edit box label changes to ID it will cause the checkpoint that checks that the edit box's label is Name to fail. Your application has changed and you need to update your component to reflect those changes. Update Run Mode enables you to update the checkpoint to reflect the change in the application.

## Running Components with the Maintenance Run Wizard

Maintenance Run Mode enables you to update your component to reflect changes in the application you are testing.

When you run a component in Maintenance Run Mode, QuickTest runs your component, and then guides you through the process of updating your steps and object repository. It does this each time it encounters a step it cannot perform due to a discrepancy between the property values of the object in the object repository and the property values of the object in the test.

When you run a component in Maintenance Mode, the Maintenance Run wizard opens for steps that failed because an object was not found in the application. You have the choice of using the wizard to point to the object in the application that you want your component to use or adding a comment to your component before the failed step. If you point to an object in the application being tested, Maintenance Mode will compare 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, Maintenance Mode will suggest one of a several options for updating your component to reflect the changes in the application. At each point in the wizard you can click the **Reset** button and point to a different object from the application for use in the failed step.

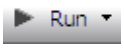
When the Maintenance Run Mode ends, Maintenance Mode wizard provides a summary of the changes it made to your component. The main Test Results window also contains a Maintenance Summary which displays details of the changes made to your component, including updated and added objects, updated and commented steps, and a summary of changes to the object repository.

**Notes:**

- You must have the Microsoft Script Debugger installed to run the components in Maintenance Mode. If it is not installed, you can use the QuickTest Additional Installation Requirements Utility to install it. (Select **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Additional Installation Requirements**.)
- You can run in Maintenance Run Mode only when QuickTest is set to use the **Normal** (displays execution marker) run mode. It cannot run in **Fast** mode. For more information, see "Setting Run Testing Options" on page 591.
- You cannot run in Maintenance Run Mode on applications that do not have a user interface, such as Web services.

**Tip:** After Maintenance Run Mode finishes, you may want to reset this setting to its previous value for regular runs.You can also update individual test object descriptions from the object in your application using the **Update from Application** option in the Object Repository window or Object Repository Manager. For more information, see "Updating Test Object Properties from an Object in Your Application" on page 138.

**To run a component in Maintenance Mode:**




1. Open the component and select **Automation** > **Maintenance Run Mode** or click the down arrow next to the **Run** button on the toolbar and select **Maintenance Run Mode**. The Run dialog box opens.

2. Specify the results location and the input parameter values (if applicable) for the Maintenance Run Mode session. For more information, see "Understanding the Results Location Tab" on page 620, and "Understanding the Input Parameters Tab" on page 621.

**3** Click **OK**. The Run dialog box closes and the Maintenance Run Mode session starts. By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 23, "Viewing Run Session Results."

---

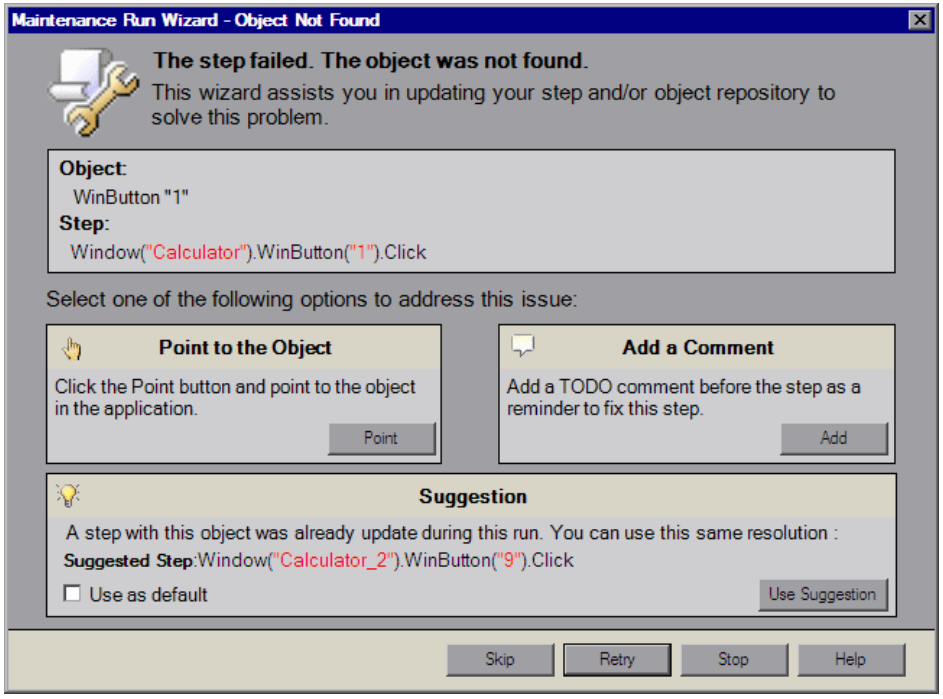
**Note:** If you cleared the **View results when run session ends** check box in the Run tab of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 20, "Setting Global Testing Options."

---

If an object in your component cannot be found in the application, the Maintenance Run Wizard opens and guides you through the steps of resolving the issue. After you resolve the issue the run continues.

## The Object Not Found Screen

If an object in your component cannot be found in the application you are testing, the Object Not Found screen opens. The Object Not Found screen identifies the **Object** that could not be found and the **Step** QuickTest was trying to perform.



---

**Notes:**

The **Suggestion** pane is displayed only if the Maintenance Run wizard cannot find an object in the application that was not found earlier in the run session as well.

The point and commenting options are disabled in the Maintenance Run wizard for objects that were not found when:

- The test is open in read-only mode.
- The object is used within a function library function.
- The object's method is defined as a registered user function.

---

The Object Not Found screen assists you in resolving the problem by providing the following options:

- **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.

  If the location to which you point is associated with several objects, the Object Selection dialog box opens. Select the correct object from the tree and click **OK**.

  One of the following screens opens depending on the object to which you pointed:

  - "The Update Step with Existing Object Screen" on page 714
  - "The Add Object to Repository Screen" on page 716
  - "The Change Object Property Values Screen" on page 711

- **Add a Comment.** Use this option if you want to add a comment to your test as a reminder to fix the failed step.

- **Suggestion.** Displayed only if Maintenance Mode cannot find an object in the application that was not found earlier in the Maintenance Mode run as well. If, when the object was first not found you chose to replace it with a different object, Maintenance Mode will suggest replacing it with the same object now.
  - **Use as default.** If, in subsequent steps the same object cannot be found, Maintenance Mode will automatically replace the object not found with the object you added to the object repository. Maintenance Mode will not open on these subsequent steps.

The Object Not Found screen contains the following navigation buttons:

- **Skip.** Skips the current step in the component and continues to run Maintenance Mode on the remainder of the component. This can be used when the problem is in the application being tested and not the QuickTest component.

---

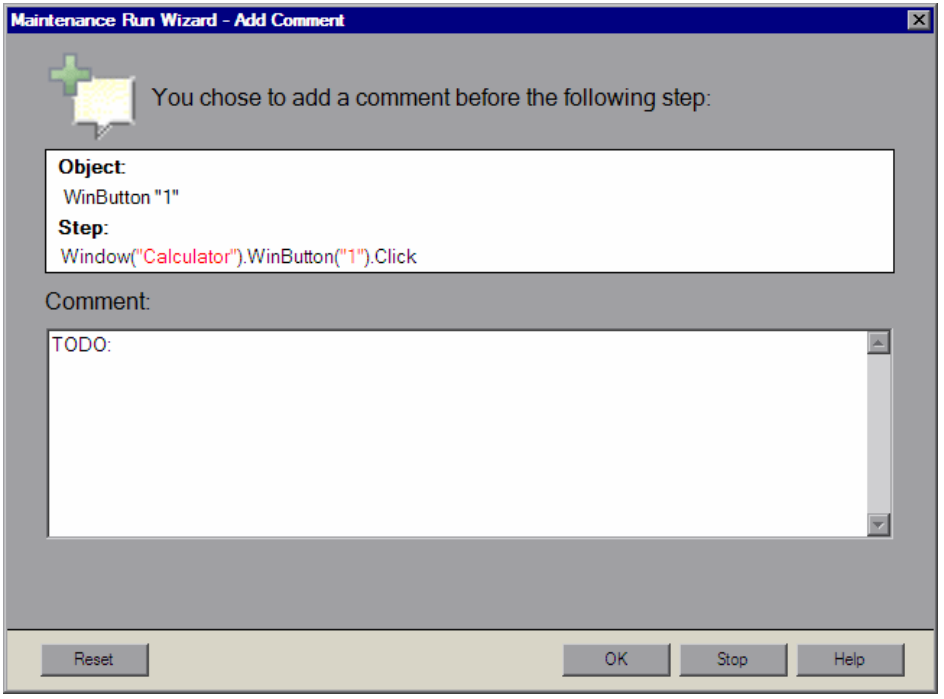
**Note:** When selecting **Skip**, ensure that the application is ready for the next step in the component.

---

- **Retry.** Retries the current step.
- **Stop.** Stops the Maintenance Run and opens the Maintenance Mode Summary screen.
- **Help.** Opens this Help topic.

## The Add Comment Screen

The Add Comment screen enables you to add a comment to your component before the current step. This can be used when identifying the object in the application will not solve the problem or you want to fix the component manually.

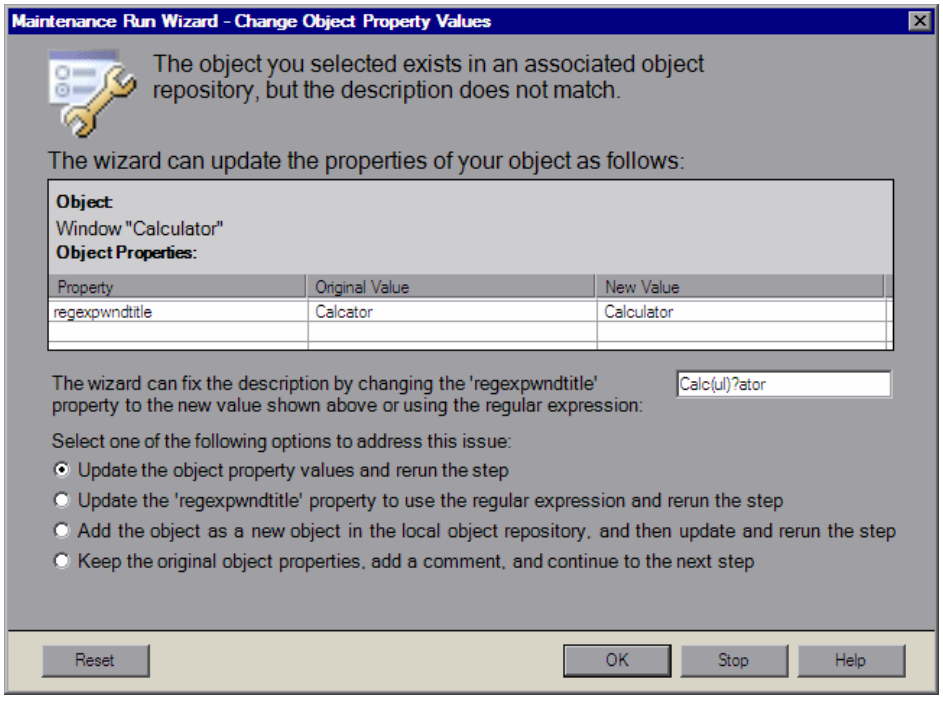



The Add Comment screen creates a comment in your component beginning with the word TODO along with text you add, as a reminder to fix the step at a later time.

## The Change Object Property Values Screen

The Change Object Property Values screen opens when an object of the same class as the object to which you pointed exists in an associated object repository, but with different description property values.

The Change Object Property Values screen suggests updating 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.




---

**Note:** If the Maintenance Run wizard does not recommend a regular expression for the new property value, the Change Object Property Value screen will not display the message and suggested regular expression below the table. The **Update the <property name> property to use the regular expression and rerun the step** radio button will also not be displayed.

---

The central area Change Object Property Values screen can contain the following information:

| Section | 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. |

Depending on the object to which you pointed, the Change Object Property Value screen may include a message that a regular expression can be used to update the property value of the object in the associated object repository. The **Update the <property name> property to use the regular expression and rerun the step** radio button will also be displayed. You can modify the suggested regular expression in the edit box. For more information on regular expressions, see "Understanding and Using Regular Expressions" on page 734.

---

**Note:** In a situation where more than one property can use a regular expression, the Maintenance Mode wizard will only suggest a regular expression for the first property value.

---

The Change Object Property Values screen provides the following options:

- **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.** Depending on the object to which you pointed one of the following screens will open:
  - The Update Step with Existing Object screen. This screen will open if the object you want to add exists in an associated object repository.
  - The Add Object to Repository screen. This screen will open if the object you want to add does not exist in an associated object repository.
- **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 screen, enabling you to add a comment before the step, and then continues to the next step.

The bottom of the screen contains the **Reset** button which enables you to return to the Object Not Found screen, where you can point to a different object in the application or choose a different course of action for this step.

---

**Notes:**

- 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 screen 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.
- Maintenance Mode makes changes to the local object repository only. If you want the new object to appear in a shared object repository, use the Object Repository Manager. For more information, see "Performing Merge Operations" on page 241.

---

## The Update Step with Existing Object Screen

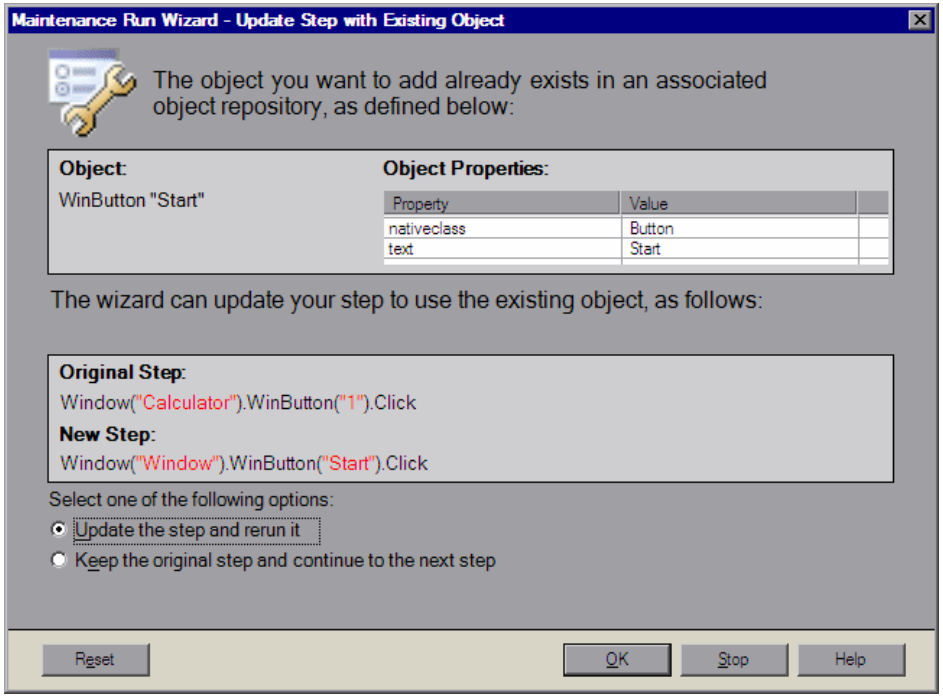
The Update Step with Existing Object screen opens if the object to which you pointed in the Object Not Found screen exists in an associated object repository and:

- No object of the same class as the object to which you pointed exists in an associated object repository with different description property values.

  Or

- In the Change Object Property Values screen you chose **Add the object as a new object in the local object repository, and then update and rerun the step**.

The Update Step with Existing Object screen suggests updating the step in your test to use an object that already exists in an associated object repository.




The central area of the Update Step with Existing Object screen contains the following sections:

| Section | 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. |

| Section | Description |
|---|---|
| **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. |

The Update Step with Existing Object screen provides the following options:

➤ **Update the step and rerun it.** Updates the failed step as shown under **New Step** and reruns the step.

---

**Note:** Maintenance Mode does not remove the original step from your 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 Maintenance Mode on the remainder of the component.

The bottom of the screen contains the **Reset** button which enables you to return to the Object Not Found screen, where you can point to a different object in the application or choose a different course of action for this step.

## The Add Object to Repository Screen

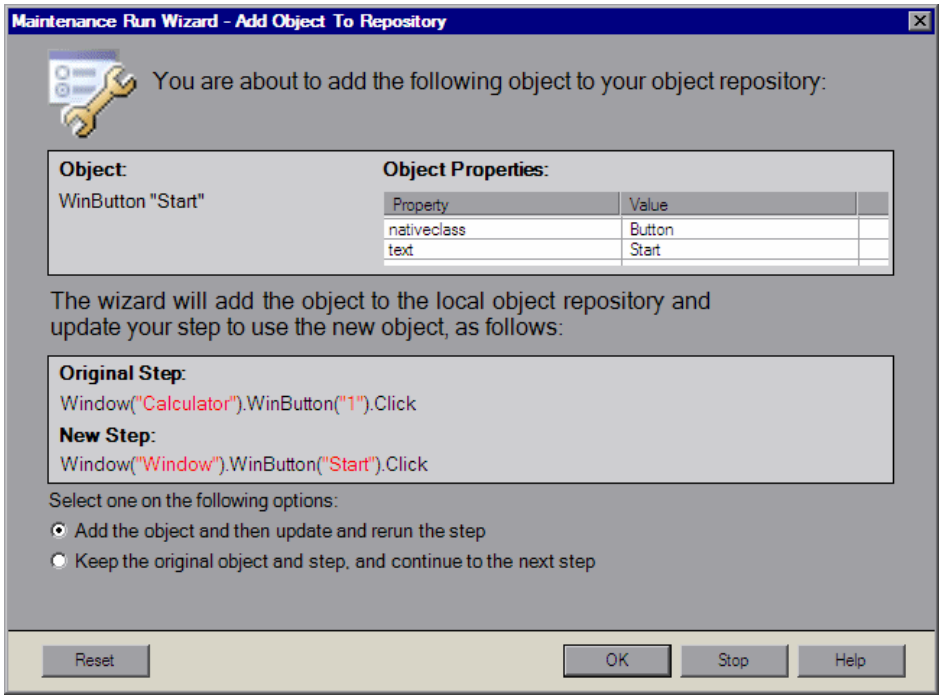
The Add Object to Repository screen opens if the object to which you pointed does not exist in any associated object repository and:

➤ No object of the same class as the object to which you pointed exists in an associated object repository with different description property values.

Or

➤ In the Change Object Property Values screen you chose **Add the object as a new object in the local object repository, and then update and rerun the step**.

The Add Object to Repository screen suggests adding the object to which you pointed to the object repository.




The central area of the Add Object to Repository screen contains the following sections:

| Section | 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. |

The Add Object to Repository screen provides the following options:

- **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.
- **Keep the original object and step, and continue to the next step.** Keeps the original step containing the original object and continues to run Maintenance Mode on the remainder of the component.

The bottom of the screen contains the **Reset** button which enables you to return to the Object Not Found screen, where you can point to a different object in the application or choose a different course of action for this step.

---

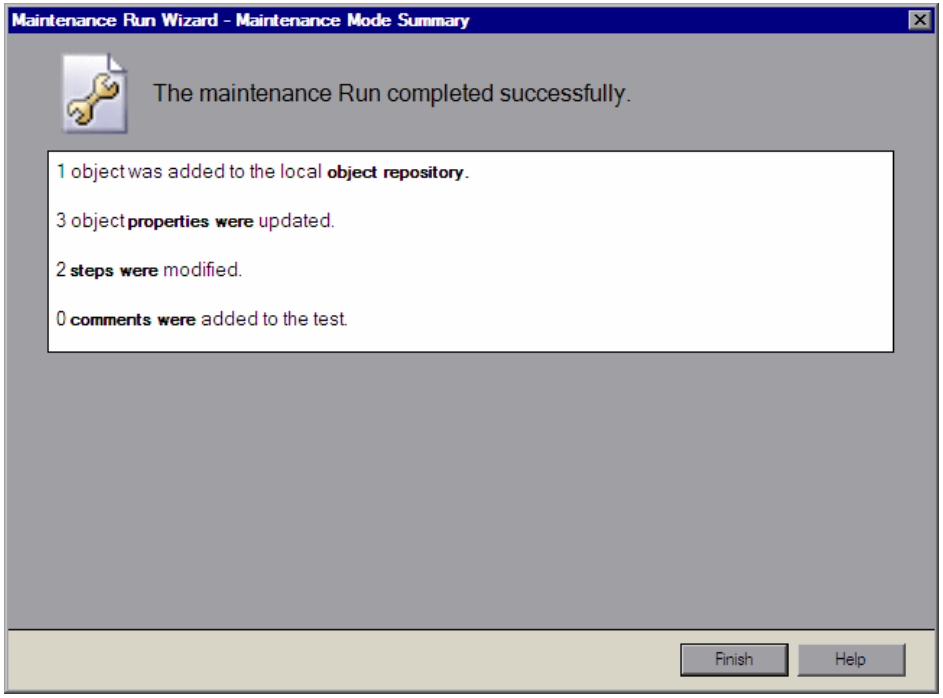
**Notes:**

- Maintenance Mode makes changes to the local object repository only. If you want the new object to appear in a shared object repository use the Object Repository Manager. For more information, see "Performing Merge Operations" on page 241.
- Maintenance Mode does not remove the original step from your component. The original step is converted into a comment and the updated step is added below it.

---

## Understanding the Maintenance Mode Summary Screen

When Maintenance Run Mode is finished, the Maintenance Mode Summary screen opens.




The Maintenance Mode Summary Screen 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.

Click **Finish** to end the Maintenance Run. By default, when the run session ends, the Test Results window opens and includes details about the steps and objects that were updated during the run. For more information on viewing the run session results, see Chapter 23, "Viewing Run Session Results."

---

**Note:** If you cleared the **View results when run session ends** check box in the Run tab of the Options dialog box, the Test Results window does not open at the end of the run session. For more information on the Options dialog box, see Chapter 20, "Setting Global Testing Options."

---

## Updating a Component Using the Update Run Mode Option

When you run a component in Update Run mode, QuickTest runs the component to update the test object descriptions and/or the expected checkpoint values. After you save the component, the updated data is used for subsequent runs.

---

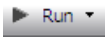
**Note:** When QuickTest updates components, it always saves the updated objects in the local object repository, even if the objects being updated were originally from a shared object repository. The next time you run the component, QuickTest uses the objects from the local object repository, as the local object repository has a higher priority than any shared object repositories.

---

---

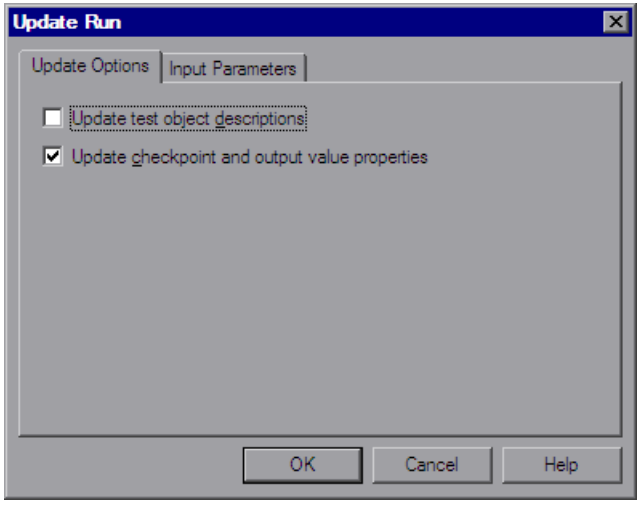
**Tip:** After using **Update Run Mode** to update the component, you may want to use the **Update from Local Repository** option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For more information, see Chapter 6, "Managing Object Repositories."

---




**1** Open the test, and select Choose **Automation** > **Update Run Mode**, or click the down arrow next to the **Run** button on the toolbar and select **Update Run Mode**.

The Update Run dialog box opens.




**2** Specify the settings for the update run process. For more information, see "Understanding the Update Options Tab" on page 723, and "Understanding the Input Parameters Tab" on page 621.

---

**Note:** The run results for an update run session are always saved in a temporary location.

---

**3** Click **OK**. The Update Run dialog box closes and QuickTest begins running in Update Run mode. The text **Update Run** flashes in the status bar while the component is being updated.

QuickTest runs the component and updates the test object descriptions and/or the expected checkpoint values, depending on your selections. When the run session ends, the Test Results window opens.

For information on viewing the results, see Chapter 23, "Viewing Run Session Results."

---

**Note:** If you cleared the **View results when run session ends** check box in the Run tab of the Options dialog box, the Test Results window does not open at the end of the update run session. For more information on the Options dialog box, see Chapter 20, "Setting Global Testing Options."

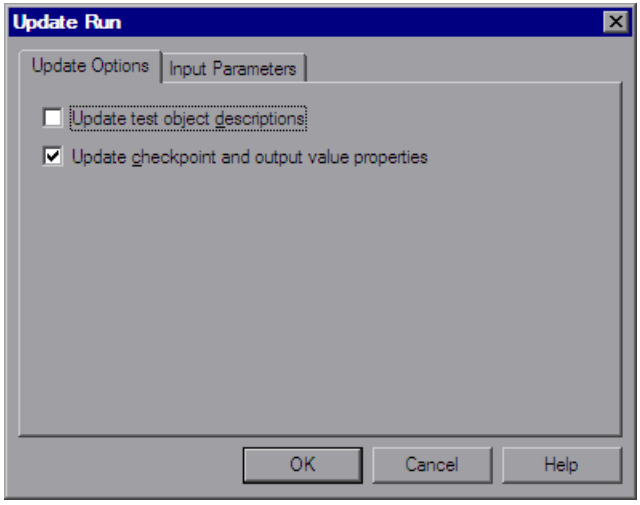
---

When the update run ends, the Test Results window can show:

- Updated values for checkpoints.
- Updated test object descriptions.

## Understanding the Update Options Tab

The Update Options tab enables you to specify which aspects of your component you want to update, such as test object descriptions and/or expected checkpoint values. After you save the component, the results of the updated component are used for subsequent runs.




You can specify one or more of the following information types to update:

- **Update test object descriptions.** QuickTest updates the set of properties for each object class in your associated object repositories according to the properties currently defined in the Object Identification dialog box. You can use this option to modify the set of properties used to identify an object of a certain type.

---

**Note:** If the property set you select in the Object Identification dialog box for an object class is not ideal for a particular object, the new object description may cause future runs to fail. Therefore, it is recommended that you save a copy of your object repository before updating it, so that you can return to the previously saved version, if necessary.

---

This option can be especially useful when you want to record and debug your component using 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 component, you can use the **Update Run Mode** option to change the object descriptions to use more universal property values.

For example, suppose you design a component for the English version of a part of your application. The test objects are recognized according to the test object property values in the English version, some of which may be language dependent. You now want to use the same component for the French version of this part of your application.

To do this, you define properties that are non-language dependent. These properties will be used 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 will modify the test object descriptions so that you can later run the component successfully on the French version of your application.

---

**Tip:** If you have a 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 component with **Update test object descriptions** selected, QuickTest finds the test object specified in each step based on its current test object description. If QuickTest cannot find the test object based on its description, it uses the Smart Identification properties to identify the test object (if Smart Identification is enabled). After QuickTest finds the test object, it then updates its description based on the mandatory and assistive properties that you define in the Object Identification dialog box.

---

**Note:** 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 Test Results. In these situations, you may want to use Maintenance Mode to resolve these problems.

---

Any properties that were used in the previous test object description and are no longer part of the description for that test object class, as defined in the Object Identification dialog box, are removed from the new description, even if the values were parameterized or defined as regular expressions.

If the same property appears both in the test object's new and previous descriptions, one of the following occurs:

- If the property value in the previous description is parameterized or specified as a regular expression and matches the new property value, QuickTest keeps the property's previous parameterized or regular expression value. For example, if the previous property value was defined as the regular expression button.*, and the new value is button1, the property value remains button.*.
- If the property value in the previous description does not match the new property value, but the object is found using Smart Identification, QuickTest updates the property value to the new, constant property value. For example, if the previous property value was button.*, and the new value is My button, if a Smart Identification definition enabled QuickTest to find the object, My button becomes the new property value. In this case, any parameterization or use of regular expressions is removed from the test object description.

➤ **Update checkpoint properties and output property values.** QuickTest updates the expected values of your checkpoints to reflect any changes that may have occurred in your application since you created the test and updates the list of items that can be retrieved in your output value steps.

For example, 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.

The output value option is mainly relevant for XML output value steps used with Web services component. For more information, see the section describing Web services in the *HP QuickTest Professional Add-ins Guide*.

---

**Notes:**

➤ If checkpoint property values are parameterized or include regular expressions, they are not updated when using this option.

➤ If you selected the **Save only selected area** check box when creating a bitmap checkpoint, the **Update Run Mode** option updates only the saved area of the bitmap; it does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint. For more information, see "Checking Bitmaps" on page 565.

---

# Part VIII

## Working with the QuickTest IDE

# 27

# QuickTest Window Layout

This chapter describes how to customize the QuickTest window and work with QuickTest documents.

**This chapter includes:**



## Modifying the QuickTest Window Layout

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

You can also resize the QuickTest window to suit your needs for each type of QuickTest session (view/edit, record, and run sessions). For example, you can display QuickTest in full view when creating or editing a component or application area, and minimize the QuickTest window during a run session. For more information, see "Customizing the QuickTest Window Layout" on page 587.

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

## Moving Panes

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

While dragging a pane, markers are displayed on the QuickTest window. If you hold the cursor over one of these markers, the area represented by the marker is shaded, enabling you to preview the window layout if the pane is moved to the selected position.

---

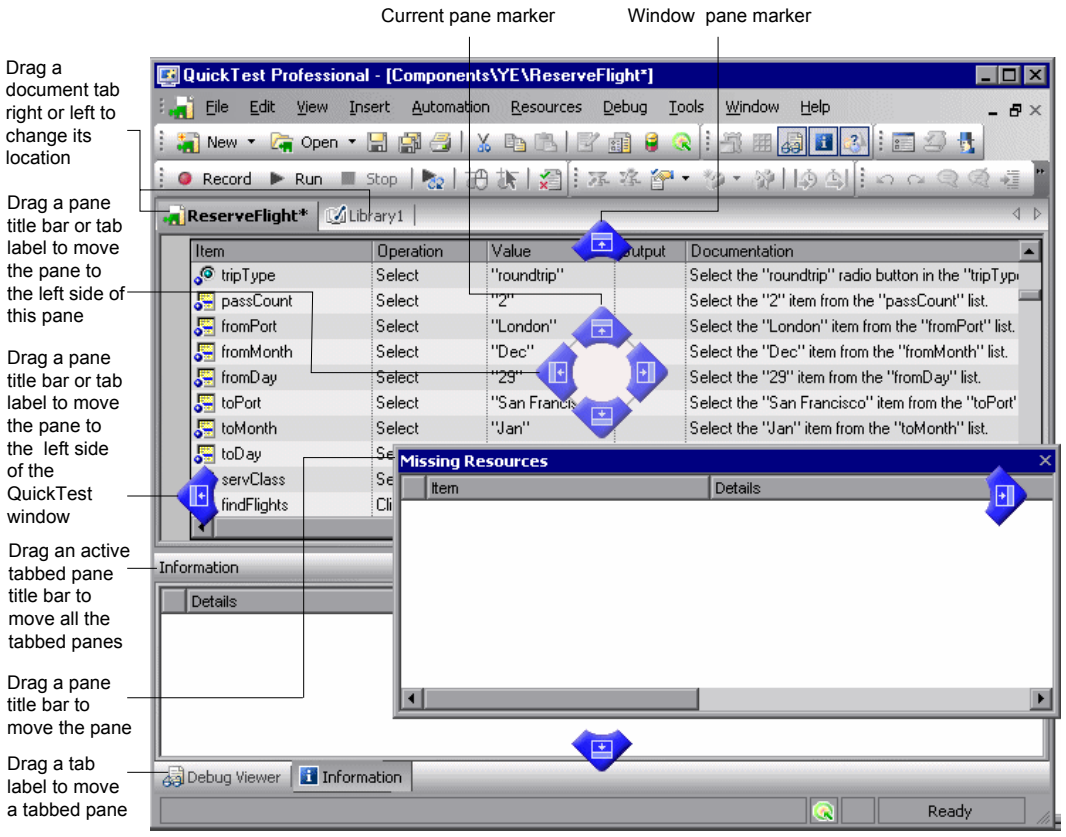
**Tip:** To move a dockable pane without snapping it into place, press CTRL while dragging it to the required location.

---

**To move panes:**

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

For example, you can move the Missing Resources pane located in the middle of the window to be a new tabbed pane at the bottom of the window. As you drag the pane, markers are displayed in the active pane and on each edge of the QuickTest window.



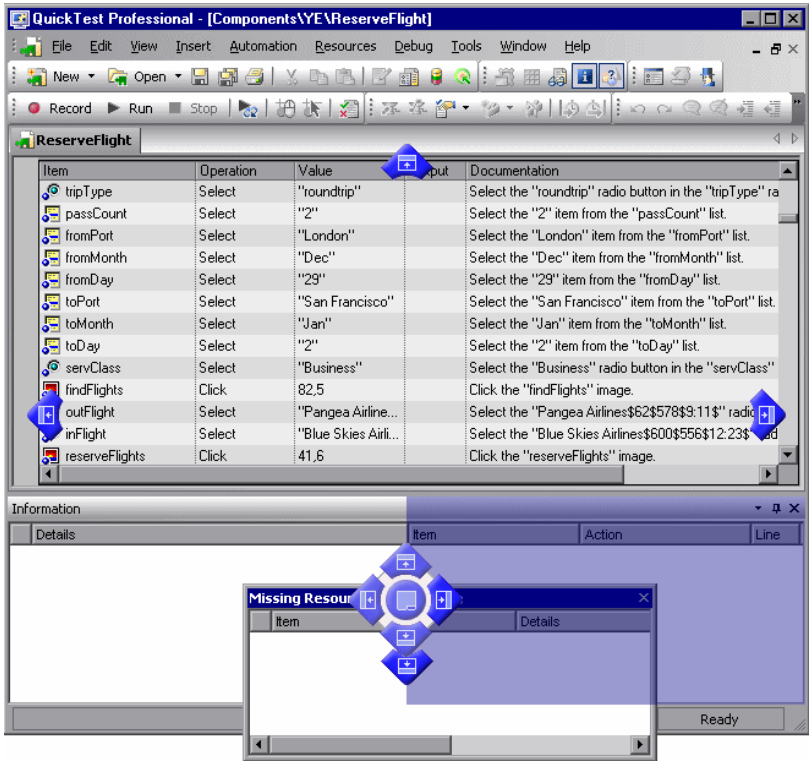

**Tips:**

- To move a single tabbed pane, drag the tab label. Once you start dragging the tabbed pane, the tab is removed, and its title bar is displayed.
- To move all the tabbed panes, drag the title bar of the active tabbed pane.

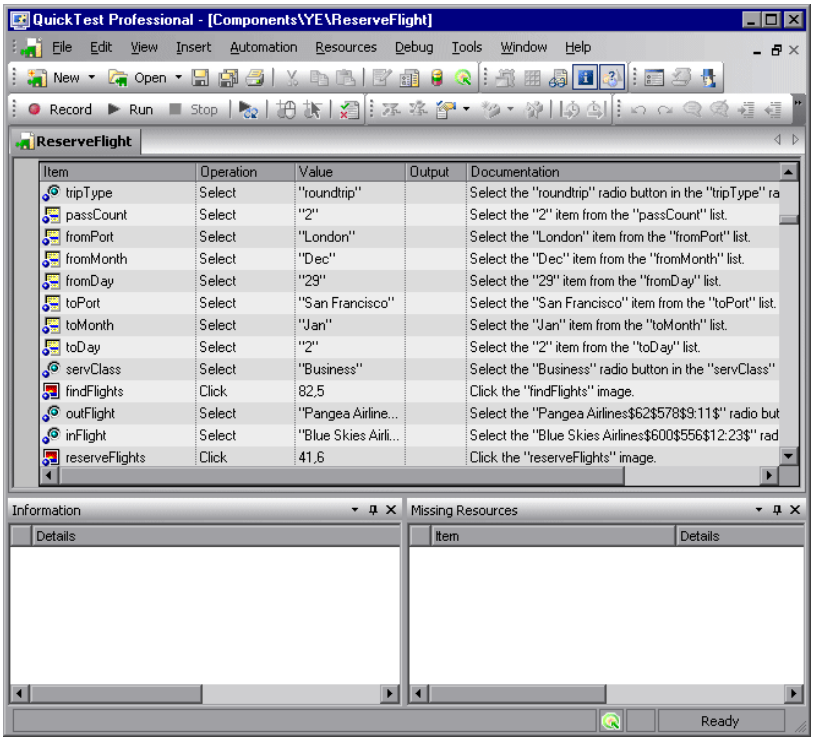
The following markers are displayed:

| Type | Marker | Description |
|---|---|---|
| **Current pane markers** |  | Enables you to:<br>➤ 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.<br>➤ position the pane as a new tabbed pane in the active window, by releasing the mouse button while selecting the center marker.<br><br>**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 QuickTest window. |
| |  | Enables you to position the pane across the right side of the QuickTest window. |
| |  | Enables you to position the pane across the bottom of the QuickTest window. |
| |  | Enables you to position the pane across the left side of the QuickTest window. |

**2** Drag the Missing Resources pane and hold the cursor over the active pane right-arrow marker, as shown below. A shaded area is displayed, indicating the new location of the pane, as shown below.



**3** Release the mouse button. The Missing Resources pane snaps into place and is displayed as a new pane in the shaded area.




---

**Tip:** You can also leave the pane as a floating pane anywhere on the QuickTest window, or on your screen. For more information on floating panes, see “Showing and Hiding Panes” on page 735.

---

**4** Repeat this procedure for each pane you want to move.

## Showing and Hiding 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.

Panes can have one of the following states—docked or floating:

➤ **Docked 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.

You can decide whether to continuously display the docked panes in the QuickTest window, or to auto-hide them. Auto-hiding means that the pane is displayed as a side-tab on the edge of the QuickTest window, and is displayed only when you hold the cursor over the tab. After you select a different pane or side-tab, the auto-hidden pane closes and is displayed as a side-tab.

---

**Note:** If you auto-hide the Information pane, it is automatically displayed when syntax errors are detected in a test script.

---

By default, auto-hidden panes open across the QuickTest window, according to their position in the QuickTest window. For example, if the docked pane was positioned on the right side of the QuickTest window, it is displayed as a side tab on the right edge of the QuickTest window, and is displayed across the right side of the QuickTest window when selected.

---

**Tip:** To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and choose **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the QuickTest window, and each pane is displayed only when you hold the cursor over that side-tab.

---

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

---

**Note:** You cannot auto-hide floating panes or individual tabbed panes.

---

**To show or hide panes:**

In the QuickTest window, select the pane you want to auto-hide, and display as a side-tab on one of the edges of the QuickTest window. The following buttons may be displayed on the title bar:

| Button | Description |
|---|---|
|  | The **Menu** button enables you to select any of the following:<br>➤ **Floating.** Opens the pane on top of all the other windows and panes, with its own title bar<br>➤ **Docking.** Docks the pane to the QuickTest window.<br>➤ **Auto-hide.** Displays the pane as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window.<br>➤ **Hide.** Closes the pane. |
|  | The **Auto Hide** button hides the pane.<br>The pane is displayed as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window.<br>To display the pane, hold the cursor over the side-tab. The button toggles to the **Dock** button, shown below. |

| Button | Description |
|---|---|
|  | The **Dock** button docks the pane to the QuickTest window.<br>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.<br>The pane is removed from the QuickTest window. To reopen the pane, select it from the **View** menu.<br>**Tip:** You can also close a pane by right-clicking and choosing **Hide** from the context menu. |

**Tips:**

- To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and choose **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the QuickTest window, and each pane is displayed only when you hold the cursor over that side-tab.
- You can float a pane by 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 choose **Docking**. The pane returns to its previous position in the QuickTest window.

### Showing and Hiding Toolbars

You can show or hide toolbars using the **View** > **Toolbars** menu options.

You can float a toolbar by moving your cursor over the toolbar handle on the left of the toolbar and then dragging the toolbar to the required position. The toolbar is displayed with a title bar.

You can double-click the title bar of the menu to dock the menu and return it to its previous position in the QuickTest window, or you can close it by clicking the **Close** button.

### Restoring the Default Layout of the QuickTest Window

You can restore the default QuickTest window layout for all document types at any time.

**To restore the default layout:**

1. Choose **Tools** > **Options**. The Options dialog box is displayed.
2. In the General tab, click the **Restore Layout** button. The panes and toolbars of all document types are restored to their default size and location.

For more information on the Options dialog box, see Chapter 20, "Setting Global Testing Options."

## Working With Multiple Documents

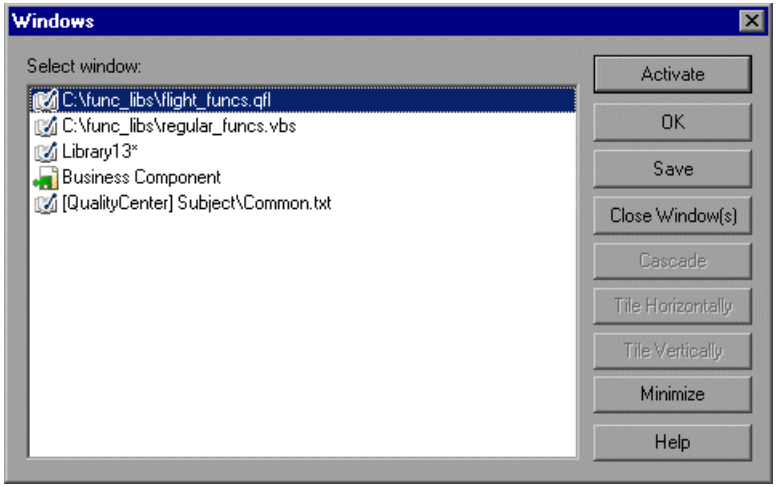
QuickTest enables you to open and work on one component or application area at a time. In addition, you can open and work on multiple function libraries simultaneously. You can open any function library, regardless of whether it is associated with the currently open component or application area.

The **Windows** menu options enable you to locate and activate (bring into focus) an open document window, select how the open document windows are arranged in the QuickTest window, or close all the open function library windows.

You can also use the Windows dialog box to manage your open QuickTest document windows.

**To work with multiple documents using the Windows dialog box:**

**1** Choose **Window** > **Windows**. The Windows dialog box opens.




The Windows dialog box displays a list of the open document windows, including the open test, component, or application area, as well as all the currently open function library windows.

**2** The Windows dialog box contains the following buttons, enabling you to manage your open documents:

| Button | Description |
|---|---|
| **Activate** | Brings the selected document into focus in the QuickTest window. |
| **OK** | Closes the Windows dialog box. |
| **Save** | Saves the selected documents. |
| **Close Window(s)** | Closes the selected function libraries. |
| **Cascade** | Arranges the selected documents in a cascading order that overlaps. |
| **Tile Horizontally** | Arranges the selected documents side-by-side horizontally, without overlapping. |
| **Tile Vertically** | Arranges the selected documents side-by-side vertically, without overlapping. |
| **Minimize** | Minimizes the selected documents. |
| **Help** | Displays the QuickTest Professional Help topic for this dialog box. |

**3** Click **OK** to close the Windows dialog box.

# 28

# Handling Missing Resources

If a component or application area has resources that cannot be found, such as missing shared object repositories, or if it uses a repository parameter that does not have a defined value, QuickTest indicates this in the Missing Resources pane. If one of the resources listed in this pane is unavailable during a run session, the test may fail. You can map a missing resource, or you can remove it from the component or application area, as required.

**This chapter includes:**

# About Handling Missing Resources

Each time you open a component or application area, QuickTest verifies that the resources specified for the component or application area are available.

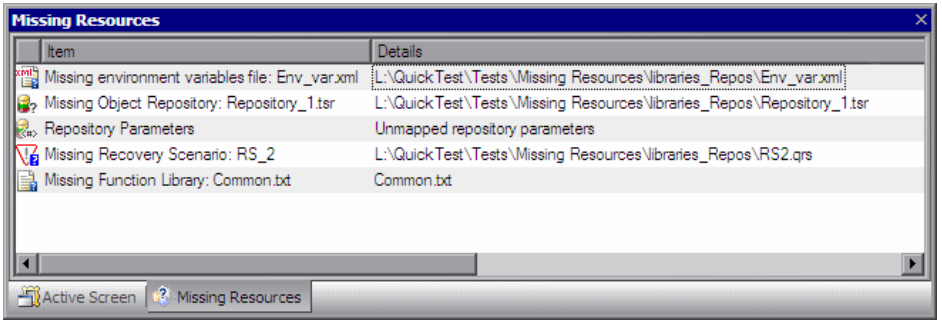
If one or more resources cannot be found, QuickTest opens the Missing Resources pane, if the pane is not already open. The Missing Resources pane provides a list of all resources that are currently unavailable, along with the location where QuickTest expected to find the resource, when available. The Missing Resources pane then enables you to locate or remove them from your application area.

---

**Note:** The Missing Resources pane does not allow missing resources in components to be resolved from the keyword view. With the exception of unmapped repository parameters, all missing resources must be resolved through the application area.

---

After you successfully handle a missing resource, QuickTest removes it from the pane.



The Missing Resources pane may list any of the following types of missing resources:

- **Missing environment variable file**. If a component or application area loads user-defined environment variables from an external file that cannot be found, QuickTest specifies the path it uses to search for the missing XML file. For more information see, "Handling Missing Environment Variables Files" on page 744.

- **Missing function library**. If a component or application area is associated with a function library that cannot be found, QuickTest specifies the path it uses to search for the missing function library. For more information see, "Handling Missing Function Libraries" on page 746.

- **Missing object repository.** If a component or application area is associated with a shared object repository that cannot be found, QuickTest specifies the path it uses to search for the missing object repository. For more information, see "Handling Missing Shared Object Repositories" on page 748.

- **Missing recovery scenario**. If a component or application area is associated with a recovery scenario that cannot be found, QuickTest specifies the path it uses to search for the missing function library. For more information see, "Handling Missing Recovery Scenarios" on page 750.

- **Repository parameters.** If a component or application area has at least one test object with a property value that is parameterized using a repository parameter that does not have a default value, QuickTest adds this generic item to the Missing Resources pane. For more information, see "Handling Unmapped Shared Object Repository Parameter Values" on page 753.

---

**Note:** In the various screens where a missing resource is used (for example, the keyword view and test settings) QuickTest indicates that a resource is missing with a special icon or text.

---

## Filtering the Missing Resources Pane

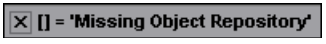
You can choose to display all missing resources in the Missing Resources pane, or only one type of missing resource.

**To filter the list of displayed missing resources:**

Right-click in the Missing Resources pane and choose one of the following:

- **All.** Displays a list of all missing resources in your component or application area.
- **Environment Variable File.** Displays the external XML file QuickTest uses to store user-defined environment variables.
- **Function Libraries.** Displays a row for each function library that cannot be found, specifying the path QuickTest uses to search for the function library.
- **Object Repositories.** Displays a row for each shared object repository that cannot be found, specifying the path QuickTest uses to search for the shared object repository.
- **Recovery Scenarios.** Displays a row for each recovery scenario that cannot be found, specifying the path QuickTest uses to search for the recovery scenario.
- **Repository Parameters.** Displays a generic row indicating that at least one test object in the repository has at least one property value that uses a repository parameter that does not have a default value.

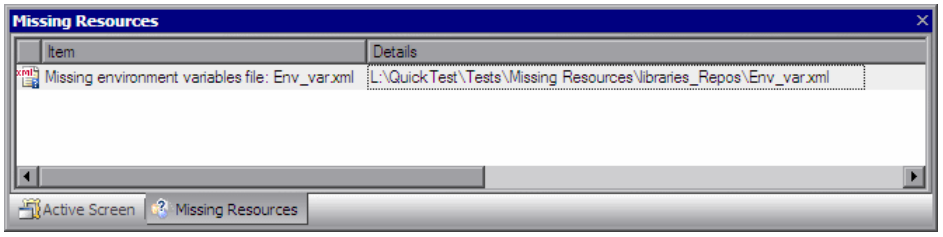
The Missing Resources pane is filtered according to the selected resource type and an indication of the applied filter is shown at the bottom of the pane:




You can cancel the filter and show all missing resources again by clicking the ✕ icon on the left of the filter indication.

# Handling Missing Environment Variables Files

When you open a component or application area that uses an external environment variables file, QuickTest verifies that the file is accessible. If an external environment variables file cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your component or application area.




The Missing Resources pane enables you to resolve a missing external environment variables file by locating or removing it.

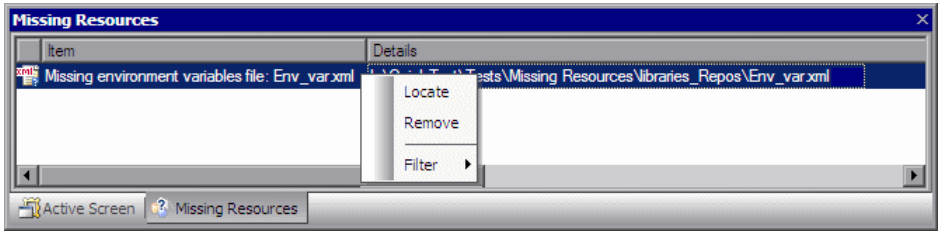
---

**Note:** The Missing Resources pane does not allow missing resources in components to be resolved from the keyword view. With the exception of unmapped repository parameters, all missing resources must be resolved through the application area.

---

**To locate a missing external environment variables file:**

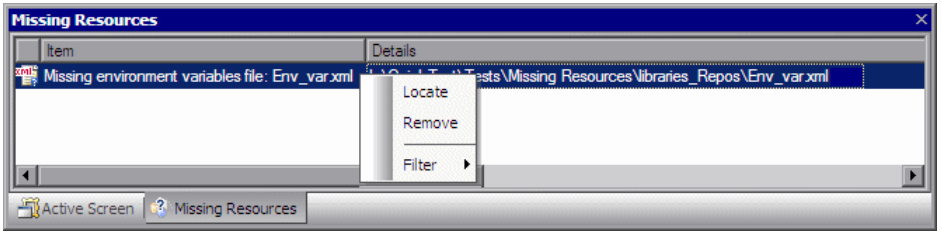
1 Right-click the missing environment variable file you want to locate and select **Locate** from the context-sensitive menu or double-click the missing environment variable file you want to locate.



The Locate Environment Variable File dialog box opens.

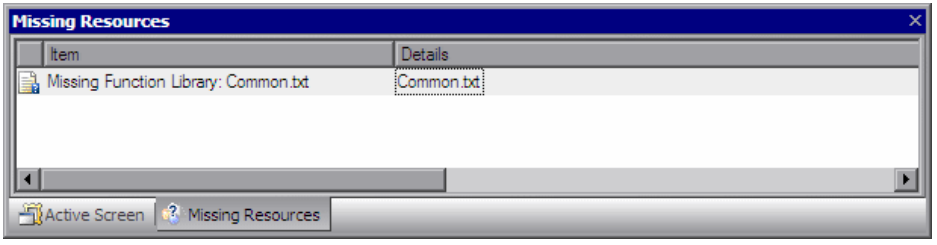
2 Browse to the environment variable file you want to use with your test and click **Open**. The selected environment variable file is used with your test and the missing environment variable file is removed from the Missing Resources pane.

To remove a missing environment variable file, right-click the missing environment variable file you want to remove and select **Remove** from the context-sensitive menu. A confirmation dialog is displayed. Click **OK** to remove the missing environment variable. The missing environment variable file is removed from your test and from the Missing Resources pane.




## Handling Missing Function Libraries

When you open a component or application area that has associated function libraries, QuickTest verifies that the libraries you specified are accessible. If a function library cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your component or application area.



The Missing Resources pane enables you to resolve a missing function library by locating or removing it.

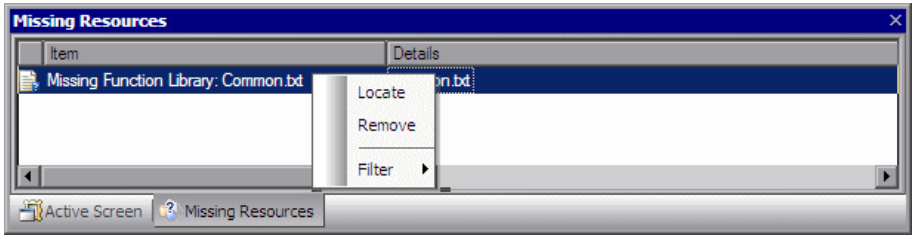
---

**Note:** The Missing Resources pane does not allow missing resources in components to be resolved from the keyword view. With the exception of unmapped repository parameters, all missing resources must be resolved through the application area.

---

**To locate a missing function library:**

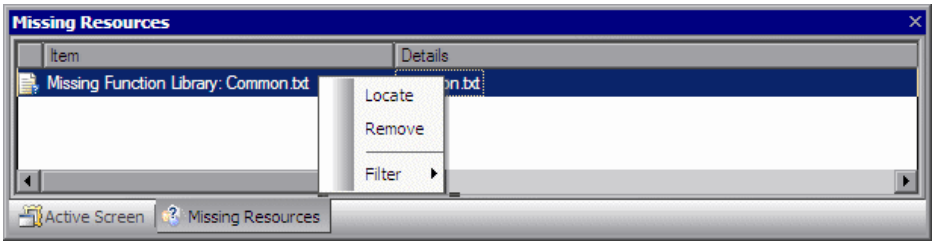
1 Right-click the missing function library you want to locate and select **Locate** from the context-sensitive menu or double-click the missing function library you want to locate.




The Locate Library dialog box opens.

2 Browse to the function library you want to associate with your test and click **Open**. The selected function library is associated with your test and the missing function library is removed from the Missing Resources pane.

To remove a missing function library, right-click the missing function library you want to remove and select **Remove** from the context-sensitive menu. A confirmation dialog is displayed. Click **OK** to remove the function library. The missing function library is removed from your test and from the Missing Resources pane.



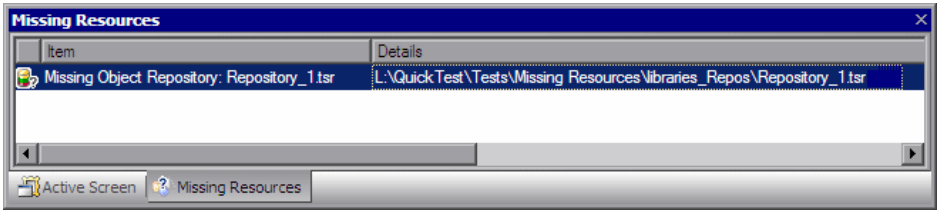

---

**Note:** When a function library is removed from your component or application area calls to those functions are not removed from your component or application area.

---

## Handling Missing Shared Object Repositories

When you associate a shared object repository with an application area, QuickTest verifies that the repository you specified is accessible. In addition, QuickTest checks that all associated shared object repositories are accessible each time you open a component or application area. If a shared object repository cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your component or application area.



For example, if you modify the name of the shared object repository or the folder in which it is stored, you will need to map the shared object repository to the associated application area.

---

**Note:** You use the Associate Repositories dialog box to resolve a missing object repository by associating a new object repository with your test. The missing object repository will still be associated with your test and will still appear in the Missing Resources pane. To remove the missing object repository from the Missing Resources pane and your test, you must use the Remove Repository feature of the Associate Repository dialog box.

---

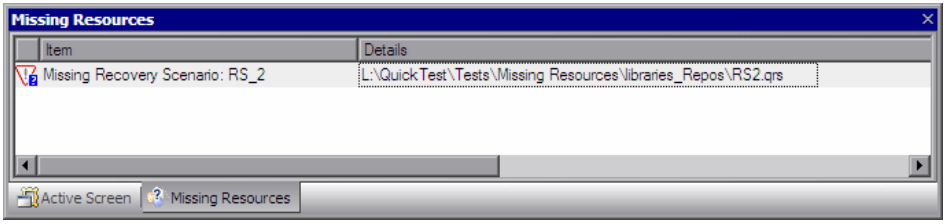
The Missing Resources pane does not allow missing resources in components to be resolved from the keyword view. With the exception of unmapped repository parameters, all missing resources must be resolved through the application area.

For a component, if you double-click the line displaying the missing object repository, QuickTest displays a message explaining that the object repository must be mapped to the associated application area. You or the Automation Engineer needs to open the application area and correct the association of the shared object repository in the Object Repositories pane.

For an application area, if you double-click the line displaying the missing object repository, QuickTest opens the Object Repositories pane of the application area, enabling you to correct the object repository association or remove it, as needed. For more information, see “Managing Shared Object Repositories” on page 432.

# Handling Missing Recovery Scenarios

When you open a component or application area that has associated recovery scenarios, QuickTest verifies that the scenarios you specified are accessible. If a recovery scenario cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your component or application area.




---

**Note:** The Missing Resources pane does not allow missing resources in components to be resolved from the keyword view. With the exception of unmapped repository parameters, all missing resources must be resolved through the application area.

---

The Missing Resources pane enables you to resolve missing recovery scenarios by:

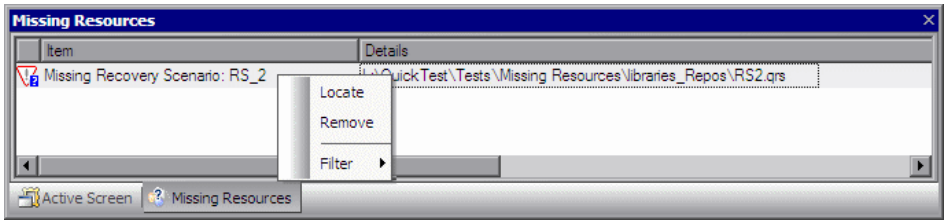
- Locating Missing Recovery Scenarios
- Removing Missing Recovery Scenarios

## Locating Missing Recovery Scenarios

The Missing Resources pane enables you to locate missing recovery scenarios in your test. If your test contains more than one missing recovery scenario, when you locate the missing scenario in a recovery file, QuickTest may identify additional missing scenarios in that file. You can instruct QuickTest to locate these missing recovery scenarios simultaneously, or you can handle each missing scenario individually.

**To locate a missing recovery scenario:**

**1** In the Missing Resources pane, right-click the recovery scenario you want to locate and select **Locate** from the context-sensitive menu or double-click the recovery scenario you want to locate.




The Locate Recovery Scenario dialog box opens.


**2** Click the Browse button to select the recovery file. The **Scenarios** area displays all the scenarios contained in the selected recovery file.

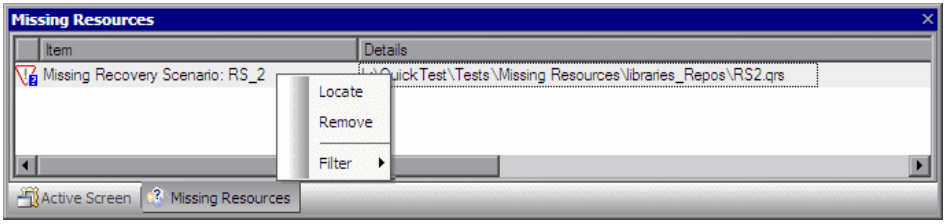
**3** Select the missing recovery scenario from the list of recovery scenarios. Click **OK**. The selected recovery scenario is associated with your test and the missing recovery scenario is removed from the Missing Resources pane.

**Note:** If your test contains additional missing recovery scenarios that can be located in the same recovery file, QuickTest opens a message box asking you if you want to map these recovery scenarios as well. Click **Yes** to map all missing recovery scenarios, or click **No** to map only the scenario you specified.

## Removing Missing Recovery Scenarios

You can remove a missing recovery scenario from a test if it is not needed.

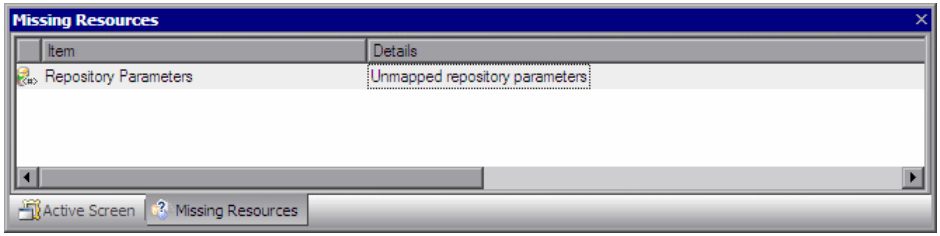
To remove a missing recovery scenario, in the Missing Resources pane, right-click the recovery scenario you want to remove and select **Remove** from the context-sensitive menu. A confirmation dialog is displayed. Click **OK** to remove the recovery scenario. The missing recovery scenario is removed from your test and from the Missing Resources pane.



# Handling Unmapped Shared Object Repository Parameter Values

Every repository parameter used in your 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 component. (For more information on repository parameters, see "Working with Repository Parameters" on page 230.)

When you open a component that uses an object repository that contains an object property whose value is parameterized using a repository parameter that does not have a value, QuickTest indicates this by displaying **Repository Parameters** in the Missing Resources pane.




For example, suppose your application contains an edit box whose name property changes depending on a selection made in a previous screen. If you parameterized the value of the name property in the object repository using a repository parameter, but a default value was not defined for the repository parameter, you need to define a value for it. You can map it to or a local or component parameter. You can also define a constant value for it, and so forth.

If you right-click the line displaying **Repository Parameters** and select **Resolve** or double-click the line displaying **Repository Parameters**, the Map Object Repository Parameters dialog box opens, enabling you to specify values for any unmapped object repository parameter. You can filter the dialog box to display only unmapped parameters or all of the parameters in the specified component (with mapped or unmapped values). For more information, see "Mapping Repository Parameter Values" on page 152.

# 29

# Adding Keywords to Your Component

QuickTest enables you to view and add the available keywords to your component in one pane.

**This chapter includes:**



## Understanding the Available Keywords Pane

The Available Keywords pane displays the keywords available to your component. It enables you to view and drag and drop objects or calls to functions into your component. When you drag and drop an object into your component, QuickTest inserts a step with the default operation for that object. When you drag and drop a function into your component, QuickTest inserts a call to that function.

For example, if you drag and drop a button object into your component, a step is added using the button object, with a click operation (the default operation for a button object).

If you drag and drop a function into your component, a comment and call to that function is added. The comment indicates that a call to the function was added to your test and indicates any necessary arguments. You need to provide the arguments for that function to your component. In the Keyword view, a tooltip displays the required arguments for the function. In the Expert view, IntelliSense displays the required arguments for the function.

You can also drag and drop test objects from other locations. For more information, see:

- "Understanding the Object Repository Window" on page 120
- "Adding Test Objects to Your Component Using the Object Repository Manager" on page 227

To view the Available Keywords pane click the Available Keywords Pane button or select **View** > **Available Keywords**.

The Available Keywords pane can display the keywords available to your component sorted by resource or sorted by keyword.

## Keywords Sorted by Resource

You can display the keywords sorted by resource by clicking the **Sort by Resource** button. Keywords are grouped by their type (library functions, local functions, objects) and then by the specific resource for that type.

- Functions in each function library are sorted alphabetically.
- Objects in each object repository are grouped by the page or window in which they appear in the application, then by the object type. They are then sorted alphabetically.
- Right-clicking a keyword enables you to open the keyword's resource or copy the selected keyword to the clipboard.
- Double-clicking a keyword opens the keyword's resource and points to the selected keyword.

## Keywords Sorted by Keyword

You can display the keywords sorted by keyword by clicking the **Sort by Keyword** button. Keywords are grouped by their type (library functions, local functions, objects) regardless of their resource.

- All available functions are sorted alphabetically.
- All available objects are grouped by the page or window in which they appear in the application, then by the object type. They are then sorted alphabetically.

---

**Note:** If two keywords have the same name, they are displayed according to the priority of their resources.

---

- Right-clicking a keyword enables you to open the keyword's resource or copy the selected keyword to the clipboard.
- Double-clicking a keyword opens the keyword's resource and points to the selected keyword.

# 30

# Managing Resources

QuickTest enables you to view and open the resources associated with your component in one pane.

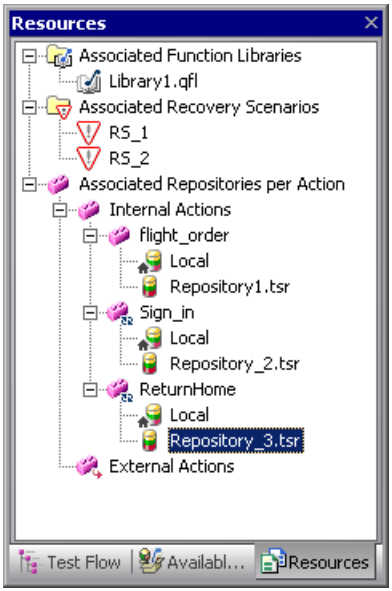
**This chapter includes:**



## Understanding the Resources Pane

Components are associated with resources such as function libraries, recovery scenarios, and object repositories. QuickTest displays many of the resources associated with a component (via its application area) in the Resources pane. The Resources pane enables you to view and open all the resources in your component.

You open the Resources pane by clicking the **Resources Pane** button, or selecting **View** > **Resources**.




The resources in the Resources pane are displayed for the current component. Function libraries and recovery scenarios are grouped by resource type. Object repositories are grouped by component.

The Resources pane is displayed as a tree structure. Right-clicking a node in the tree opens the context menu for that resource. Some options are accessible through the context menu of the root node for a resource and some options are accessible through the context menu of the specific resource.

## Associated Function Libraries

The **Associated Function Libraries** node represents all the function libraries currently associated with your test.

The context menu for the **Associated Function Libraries** root node contains:

- **Associate Function Library**. This option is disabled for components.

The context menu for a specific function library contains:

- **Open Function Library**. Displays the selected function library in a function library window in the display area. You can also double-click to open a function library.
- **Remove Function Library from List**. This option is disabled for components.
- **Move Up** or **Move Down**. This option is disabled for components.

For more information see Chapter 11, "Working with User-Defined Functions and Function Libraries."

## Associated Recovery Scenarios

The **Associated Recovery Scenarios root** node represents all the recovery scenarios currently associated with your test.

The context menu for the **Associated Recovery Scenarios** root node contains:

- **Associate Recovery Scenario**. This option is disabled for components.

The context menu for a recovery scenario contains:

- **Recovery Scenario Properties**. This option is disabled for components.
- **Remove Recovery Scenario from List**. This option is disabled for components.
- **Disable Recovery Scenario** or **Enable Recovery Scenario**. This option is disabled for components.
- **Move Resource Up** or **Move Resource Down**. This option is disabled for components.

Double-clicking a recovery scenario opens the Recovery Scenario Manager dialog box.

For more information, see Chapter 32, "Defining and Using Recovery Scenarios."

## Associated Repositories for Component

Actions are grouped under the **Associated Repository for Component** node into **Internal Actions** and **External Actions**. Displayed beneath the **Internal Actions** and **External Actions** are the local object repository and any shared object repositories associated with that component.

---

**Note:** The Resources Pane lists all the actions stored with your component even when they are not called by your component. For more information on how actions are stored with your component, see "Using the Action Toolbar in the Keyword View" on page 426.

---

The context menu for an component contains:

- **Open Repository**. This option is disabled for components.
- **Remove Repository from List**. This option is disabled for components.
- **Move Up** or **Move Down**. This option is disabled for components.

For more information on working with object repositories, see Chapter 4, "Working with Objects."

# 31

# Working with Process Guidance

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

A process is a collection of activities, or sub-processes. Each process walks you step-by-step through the activities that are required for that process. As you navigate through the activities for each process and perform the tasks described in each activity, you become acquainted with the way in which a particular process should be performed.

QuickTest provides a built-in package that comprises several processes. These processes provide introductory information and tips on how to perform the most common QuickTest tasks, such as planning and creating an application area or business component.

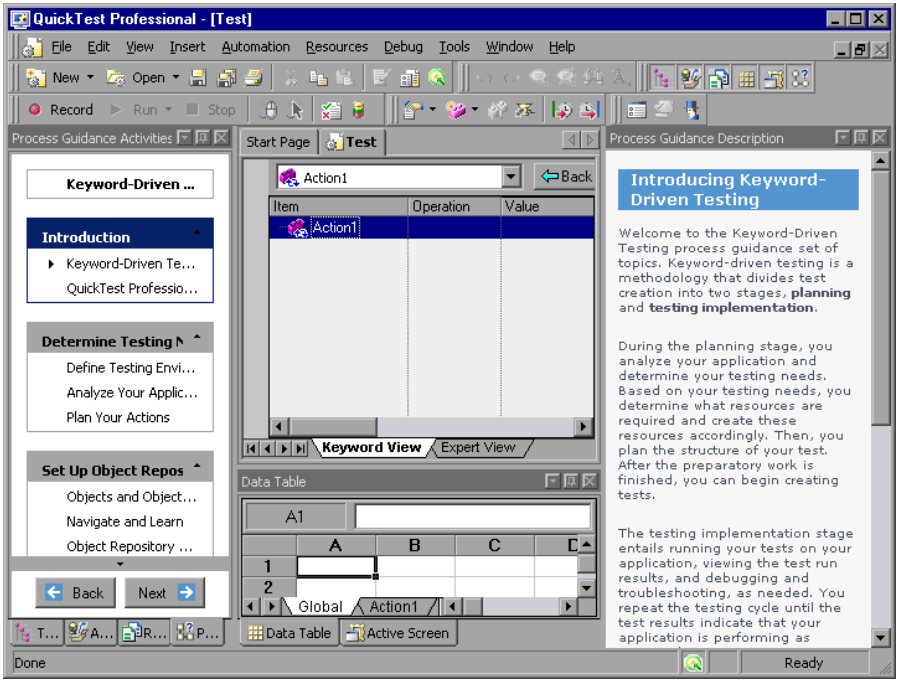
Your organization can also create its own custom processes to guide users through specific requirements and best practices relevant to your organization.

**This chapter includes:**

# Process Guidance Panes

In QuickTest, process guidance is displayed in two panes: the **Process Guidance Activities** pane and the **Process Guidance Description** pane.

You display or hide these panes by choosing **View** > **Process Guidance** or clicking the **Process Guidance panes** toggle button.




## Process Guidance Activities Pane

The **Process Guidance Activities** pane (shown on the left) lists the activities that are part of the selected process. Activities are often grouped, enabling you to navigate directly to the sub-process that interests you. The example above illustrates some of the groups and activities in the Keyword-Driven Testing process. For example, the Determine Testing Needs group contains three activities: Define Testing Environment, Analyze Your Application, and Plan Your Actions.

In the **Process Guidance Activities** pane, you can:

- Click an activity to open the relevant topic in the **Process Guidance Description** pane.
- Check which activity is displayed in the **Process Guidance Description** pane. (An arrow points to the currently selected activity.)
- Use the **Back** and **Next** buttons to navigate up and down between activities and to display the topic for the previous or next activity in the **Process Guidance Description** pane.
- Position the cursor over the **Up** and **Down** arrows to scroll through the list of activities. (The up arrow is located directly below the Process Guidance Activities title bar; the down arrow is located directly above the **Back** and **Next** buttons.)

### Process Guidance Description Pane

The **Process Guidance Description** pane (shown on the right in the example above) displays the topic (description), for the selected activity.

Each description introduces you to a specific activity and provides links to locations in which you can find more information about how to perform that activity. Additionally, many of the descriptions include interactive links that open dialog boxes or other relevant features, enabling you to directly access the features that are being described.

# Opening Process Guidance

You can open a process from the Start Page, from the **Automation** menu, or from the Process Guidance Activities pane.

## Start Page

The **Process Guidance List** on the Start Page displays all available processes. Some processes may be available only under certain conditions. For example, the Business Components process guidance is available only if you are connected to a Quality Center project that supports business process testing. Additionally, some processes may be visible only if you have a specific add-in loaded. For example, the **Testing SAP Gui for Windows** built-in process is visible only if the SAP add-in is loaded.

When you select a QuickTest process from the list, the relevant document type opens. For example, if a test document is open and you select the **Application Areas** process, a new application area opens, enabling you to navigate through the application area as you navigate through the selected process (provided that you are connected to a Quality Center project with business process testing support).

**To open a specific process from the Start Page:**

1. In QuickTest, click the **Start Page** tab to display the Start Page. (If the Start Page tab is not visible, choose **View** > **Start Page** to open the Start Page.)
2. In the **Process Guidance List**, click the link for the process you want to open. The list of activities is displayed in the **Process Guidance Activities** pane, and a description of the first activity in the list is displayed in the **Process Guidance Description** pane.

---

**Tip:** If the **Process Guidance List** is empty, choose **File** > **Process Guidance Management** and select at least one process in the Process Guidance Management dialog box.

---

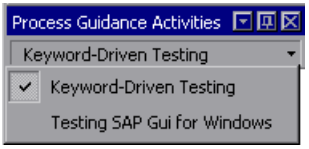
### Automation Menu Command

You can open any process that is available for a currently open document type or for a loaded QuickTest add-in by choosing **Automation** > **Process Guidance List** and then choosing a process from the list.

If the **Process Guidance List** is empty, choose **File** > **Process Guidance Management** and select at least one process in the Process Guidance Management dialog box. Then reopen the current document or open a new document to refresh the list of available processes in the **Process Guidance List** menu.

(If you want to open a process that is not relevant for the current testing document or loaded QuickTest add-in, you need to open the process from **Process Guidance List** in the Start Page.)

If the currently open testing document has more than one process available, you can navigate between these process by selecting the required process from the drop-down list in the process title.

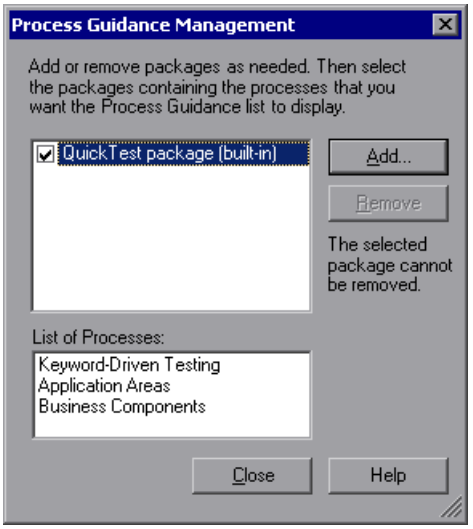



## Managing the List of Available Processes

Processes are stored in process guidance packages. QuickTest provides a built-in package containing several processes. This package is listed by default in the Process Guidance Management dialog box.

Your organization may provide additional packages that include processes that are specific to your organization, your team, your role in your organization, and so on.

You use the Process Guidance Management dialog box to manage the list of processes that are available in QuickTest.




## Including and Excluding Packages

You can select to include or exclude a package in the set of packages available in QuickTest.

When you select to include a package, QuickTest adds all of the processes in that package to the **Process Guidance List** on the Start Page (excluding processes for QuickTest add-ins that are not currently loaded). The processes that are available for the currently open document type and for the currently loaded QuickTest add-ins are also added to the **Process Guidance List** in the **Automation** menu, and can be opened after you refresh the list by closing and reopening the current document or by opening a new document of the same type.

You cannot include or exclude individual processes from within a package.

**To include or exclude a package in the set of packages available in QuickTest:**

1 Choose **File** > **Process Guidance Management**. The Process Guidance Management dialog box opens.

2 Select the check box adjacent to the package whose processes you want to include, or clear the check box adjacent to the package whose processes you want to exclude.

3 Click **Close**. QuickTest adds or removes the relevant processes in the **Process Guidance List**.

## Adding Process Guidance Packages

If your organization has its own processes, you can add them to the **Process Guidance List** on the Start Page. You do this by adding the relevant package to the Process Guidance Management dialog box and selecting to show it.

**To add a package to the list:**

1 Choose **File** > **Process Guidance Management**. The Process Guidance Management dialog box opens.

2 In the Process Guidance Management dialog box, click **Add**. The Open dialog box opens.

3 Browse to the process guidance package file and click **Open**. The package is added to the list of available packages.

# Part IX

## Working with Advanced Features

# 32

# Defining and Using Recovery Scenarios

You can instruct QuickTest to recover from unexpected events and errors that occur in your testing environment during a run session.

**This chapter includes:**

# About Defining and Using Recovery Scenarios

Unexpected events, errors, and application crashes during a run session can disrupt your run session and distort results. This is a problem particularly when components run unattended—the component pauses until you perform the operation needed to recover. To handle situations such as these, QuickTest enables you to create recovery scenarios and associate them with specific components. Recovery scenarios activate specific recovery operations when trigger events occur. For information on when to use recovery scenarios, see "Deciding When to Use Recovery Scenarios" on page 776.

The Recovery Scenario Manager provides a wizard that guides you through the process of defining a recovery scenario, which includes a definition of an unexpected event and the operations necessary to recover the run session. For example, you can instruct QuickTest to detect a **Printer out of paper** message and recover the run session by clicking the **OK** button to close the message and continue the component.

A recovery scenario consists of the following:

- **Trigger Event.** The event that interrupts your run session. For example, a window that may pop up on screen, or a QuickTest run error.
- **Recovery Operations.** The operations to perform to enable QuickTest to continue running the component after the trigger event interrupts the run session. For example, clicking an **OK** button in a pop-up window, or restarting Microsoft Windows.
- **Post-Recovery Test Run Option.** The instructions on how QuickTest should proceed after the recovery operations have been performed, and from which point in the component QuickTest should continue, if at all. For example, you may want to restart a component from the beginning, or skip a step entirely and continue with the next step in the component.

Recovery scenarios are saved in recovery scenario files. A recovery scenario file is a logical collection of recovery scenarios, grouped according to your own specific requirements.

To instruct QuickTest to perform a recovery scenario during a run session, you must first associate the recovery scenario with that component (via its application area). A component can have any number of recovery scenarios associated with it. You can prioritize the scenarios associated with your component to ensure that trigger events are recognized and handled in the required order. For more information, see "Defining Recovery Scenario Settings for Your Application Area" on page 447.

When you run a component for which you have defined recovery scenarios and an error occurs, QuickTest looks for the defined trigger events that caused the error. If a trigger event has occurred, QuickTest performs the corresponding recovery and post-recovery operations.

You can also control and activate your recovery scenarios during the run session by inserting Recovery statements into your component. For more information, see "Programmatically Controlling the Recovery Mechanism" on page 815.

---

**Note:** If you choose **On error** in the **Activate recovery scenarios** box in the Recovery tab of the Application Area Settings dialog box, the recovery mechanism does not handle triggers that occur in the last step of a component. If you chose this option and need to recover from an unexpected event or error that may occur in the last step of a component, you can do this by adding an extra step to the end of your component.

---

## Deciding When to Use Recovery Scenarios

Recovery scenarios are intended for use **only** with events that you cannot predict in advance, or for events that you cannot otherwise synchronize with a specific step in your component. For example, you could define a recovery scenario to handle printer errors. Then if a printer error occurs during a run session, the recovery scenario could instruct QuickTest to click the default button in the Printer Error message box.

You would use a recovery scenario in this example because you cannot handle this type of error directly in your 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 immediately after a step that sends a file to the printer, your component may progress several steps before the network returns the actual printer error.

If you can predict that a certain event may happen at a specific point in your component, it is highly recommended to handle that event directly within your component by adding steps such as If statements in user-defined functions, rather than depending on a recovery scenario. For example, 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 in a user-defined function that clicks **OK** if the message box opens.

Handling an event directly within your 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. 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 component.

# Defining Recovery Scenarios

You create recovery scenarios using the Recovery Scenario Wizard (accessed from the Recovery Scenario Manager dialog box). The Recovery Scenario Wizard leads you through the process of defining each of the stages of a recovery scenario. As you create your recovery scenarios, you save them in a recovery file. A recovery file is a convenient way to organize and store multiple recovery scenarios together.

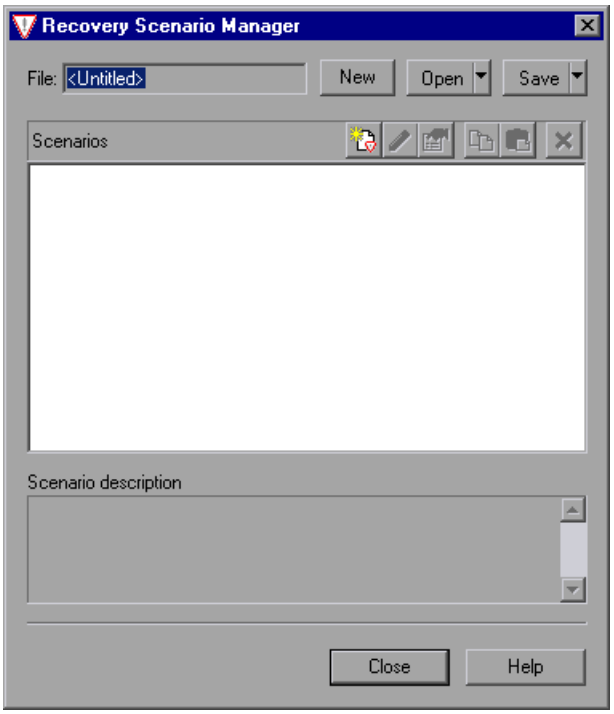
Using the Recovery Scenario Manager dialog box, you can then select any recovery file to manage all of the recovery scenarios stored in that file. This enables you to edit a selected recovery scenario, associate specific recovery scenarios with specific components to instruct QuickTest to implement the recovery scenarios when specified trigger events occur, and so forth.

## Creating a Recovery File

You create recovery files to store your recovery scenarios. You can create a new recovery file or edit an existing one.

**To create a recovery file:**

**1** Choose **Resources** > **Recovery Scenario Manager**. The Recovery Scenario Manager dialog box opens.



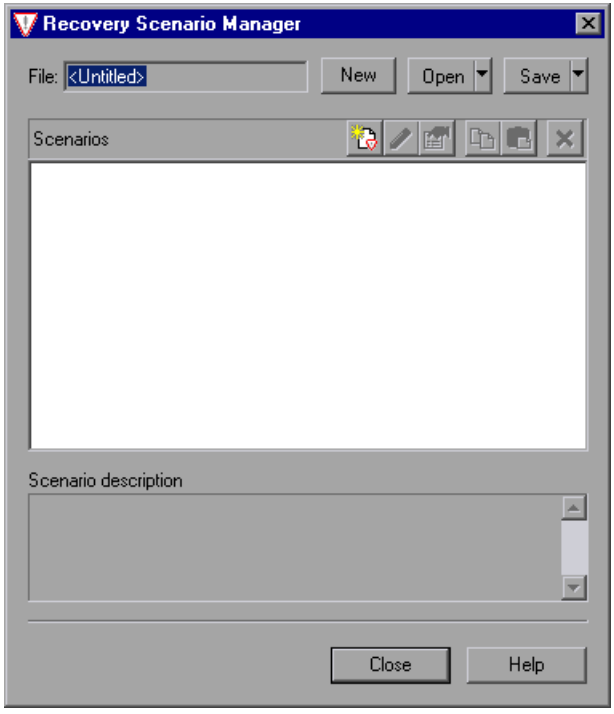

**2** By default, the Recovery Scenario Manager dialog box opens with a new recovery file. You can either use this new file, or click the **Open** button to choose an existing recovery file. Alternatively, you can click the arrow next to the **Open** button to select a recently-used recovery file from the list.

You can now create recovery scenarios using the Recovery Scenario Wizard and save them in your recovery file, as described in the following sections.

## Understanding the Recovery Scenario Manager Dialog Box

The Recovery Scenario Manager dialog box enables you to create and edit recovery files, and create and manage the recovery scenarios stored in those files.

The Recovery Scenario Manager dialog box displays the name of the currently open recovery file, a list of the scenarios saved in the recovery file, and a description of each scenario.



The Recovery Scenario Manager dialog box contains the following toolbar buttons:

| Option | Description |
|---|---|
|  | Creates a new recovery file. For more information, see “Creating a Recovery File” on page 777. |
|  | Opens an existing recovery file. You can also click the arrow to select a recovery file from the list of recently-used recovery files. |
|  | Saves the current recovery file. For more information, see “Saving the Recovery Scenario in a Recovery File” on page 805. |
|  | Opens the Recovery Scenario Wizard, in which you define a new recovery scenario. For more information, see “Understanding the Recovery Scenario Wizard” on page 781. |
|  | Opens the Recovery Scenario Wizard for the selected recovery scenario, in which you can modify the recovery scenario settings. For more information, see “Modifying Recovery Scenarios” on page 809. |
|  | Displays summary properties for the selected recovery scenario in read-only format. For more information, see “Viewing Recovery Scenario Properties” on page 807. |
|  | Copies a recovery scenario from the open recovery file to the Clipboard. This enables you to paste a recovery scenario into another recovery file. For more information, see “Copying Recovery Scenarios between Recovery Scenario Files” on page 810. |
|  | Pastes a recovery scenario from the Clipboard into the open recovery file. For more information, see “Copying Recovery Scenarios between Recovery Scenario Files” on page 810. |
|  | Deletes a recovery scenario. For more information, see “Deleting Recovery Scenarios” on page 809. |

---

**Note:** Each recovery scenario is represented by an icon that indicates its type. For more information, see “Managing Recovery Scenarios” on page 806.

---

# Understanding the Recovery Scenario Wizard

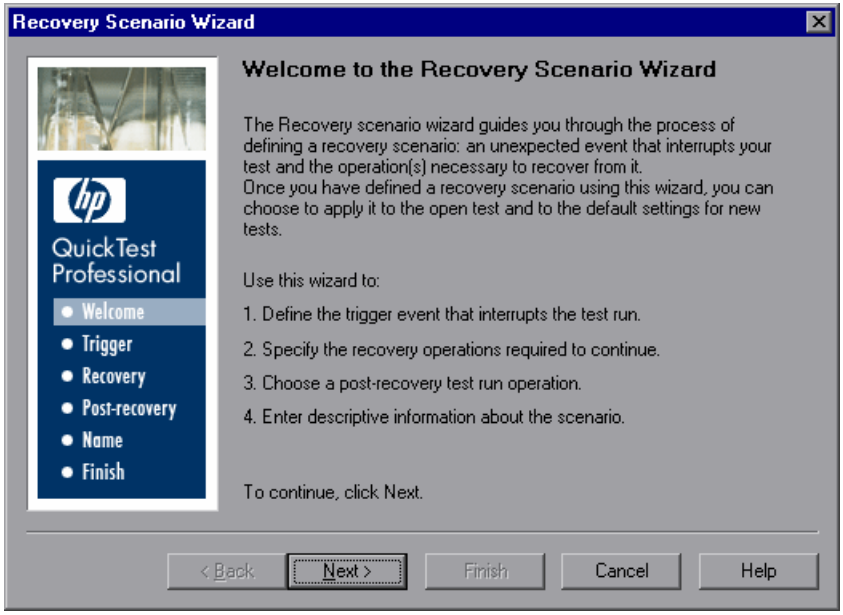
The Recovery Scenario Wizard leads you, step-by-step, through the process of creating a recovery scenario. The Recovery Scenario Wizard contains the following main steps:

- defining the trigger event that interrupts the run session
- specifying the recovery operations required to continue
- choosing a post-recovery test run operation
- specifying a name and description for the recovery scenario

You open the Recovery Scenario Wizard by clicking the **New Scenario** button in the Recovery Scenario Manager dialog box (**Resources** > **Recovery Scenario Manager**).

## Welcome to the Recovery Scenario Wizard Screen

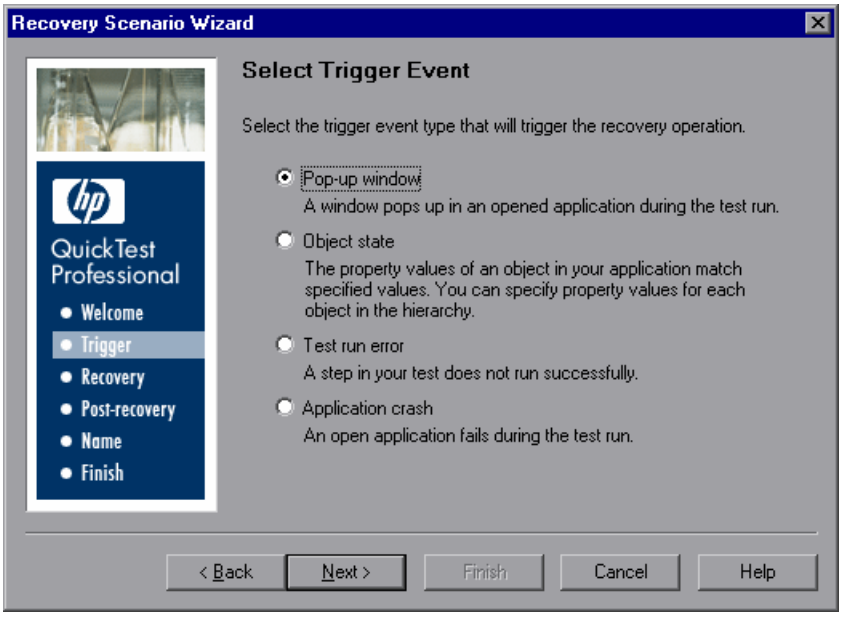
The Welcome to the Recovery Scenario Wizard screen provides general information on the different options in the Recovery Scenario Wizard, and provides an overview of the stages involved in defining a recovery scenario.



Click **Next** to continue to the Select Trigger Event Screen (described on page 782).

## Select Trigger Event Screen

The Select Trigger Event screen enables you to define the event type that triggers the recovery scenario, and the way in which QuickTest recognizes the event.




Select a type of trigger and click **Next**. The next screen displayed in the wizard depends on which of the following trigger types you select:

➤ **Pop-up window.** QuickTest detects a pop-up window and identifies it according to the window title and textual content. For example, a message box may open during a run session, indicating that the printer is out of paper. QuickTest can detect this window and activate a defined recovery scenario to continue the run session.

Select this option and click **Next** to continue to the Specify Pop-up Window Conditions Screen (described on page 784).

➤ **Object state.** QuickTest detects a specific test object state and identifies it according to its property values and the property values of all its ancestors. Note that an object is identified only by its property values, and not by its class.

For example, a specific button in a dialog box may be disabled when a specific process is open. QuickTest can detect the object property state of the button that occurs when this problematic process is open and activate a defined recovery scenario to close the process and continue the run session.

Select this option and click **Next** to continue to the Select Object Screen (described on page 786).

➤ **Test run error.** QuickTest detects a run error and identifies it by a failed return value from a method. For example, QuickTest may not be able to identify a menu item specified in the method argument, due to the fact that the menu item is not available at a specific point during the run session. QuickTest can detect this run error and activate a defined recovery scenario to continue the run session.

Select this option and click **Next** to continue to the Select Test Run Error Screen (described on page 789).

➤ **Application crash.** QuickTest detects an application crash and identifies it according to a predefined list of applications. For example, a secondary application may crash when a certain step is performed in the run session. You want to be sure that the run session does not fail because of this crash, which may indicate a different problem with your application. QuickTest can detect this application crash and activate a defined recovery scenario to continue the run session.

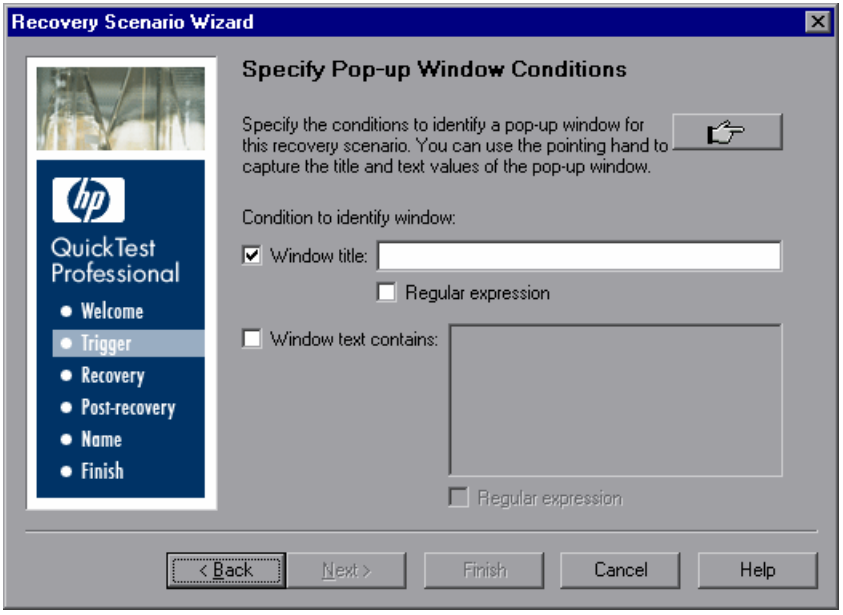
Select this option and click **Next** to continue to the Recovery Operations Screen (described on page 792).

**Notes:**

- The set of recovery operations is performed for each occurrence of the trigger event criteria. For example, suppose 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 component. If you need to recover from an unexpected event or error that may occur in the last step of a component, you can do this by adding an extra step to the end of your component.

## Specify Pop-up Window Conditions Screen

If you chose a **Pop-up window** trigger in the Select Trigger Event Screen (described on page 782), the Specify Pop-up Window Conditions screen opens.



Perform one of the following to specify how the pop-up window should be identified:

- Choose whether you want to identify the pop-up window according to its **Window title** and/or **Window text** and then enter the text used to identify the pop-up window. You can use regular expressions in the window title or textual content by selecting the relevant **Regular expression** check box and then entering the regular expression in the relevant location. For information on regular expressions, see the *HP QuickTest Professional User's Guide*.
- Click the pointing hand. Then click the pop-up window to capture the window title and textual content of the window.

---

**Note:** Using the first option (**Window title** and/or **Window text**) instructs QuickTest to identify any pop-up window that contains the relevant title and/or text. Using the second option (pointing hand) instructs QuickTest to identify only pop-up windows that match the object property values of the window you select.

---

---

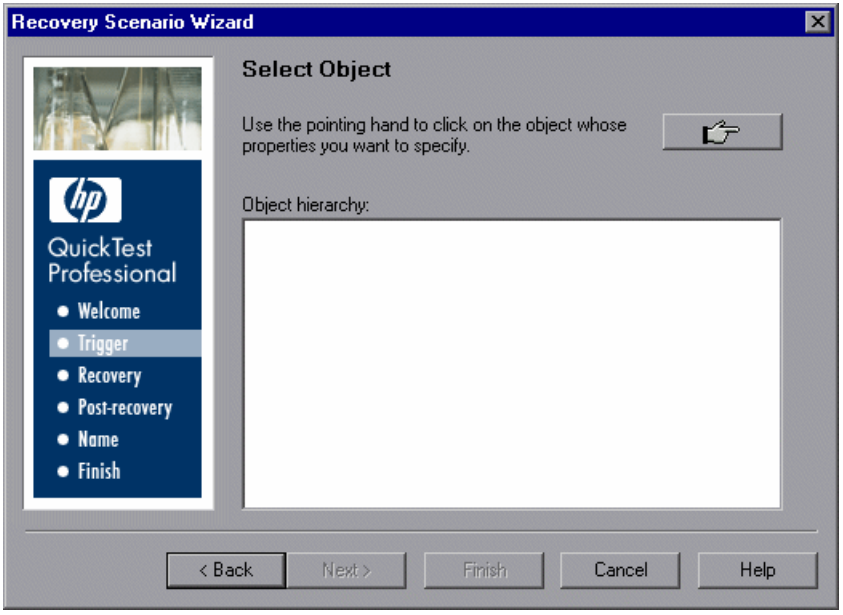
**Tip:** Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

---

Click **Next** to continue to the Recovery Operations Screen (described on page 792).

## Select Object Screen

If you chose an **Object state** trigger in the Select Trigger Event Screen (described on page 782), the Select Object screen opens.



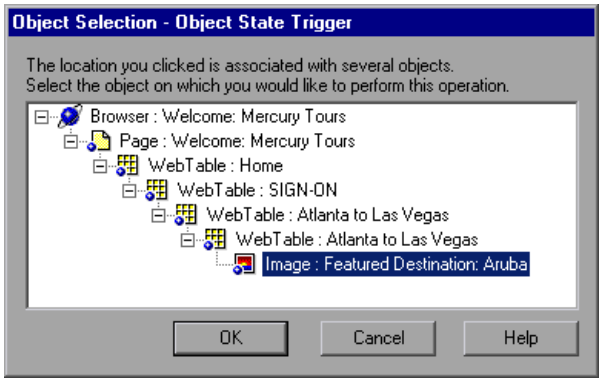

Click the pointing hand and then click the object whose properties you want to specify.

---

**Tip:** Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

---

If the location you click is associated with more than one object, the Object Selection–Object State Trigger dialog box opens.




Select the object whose properties you want to specify and click **OK**. The selected object and its parents are displayed in the Select Object screen.

---

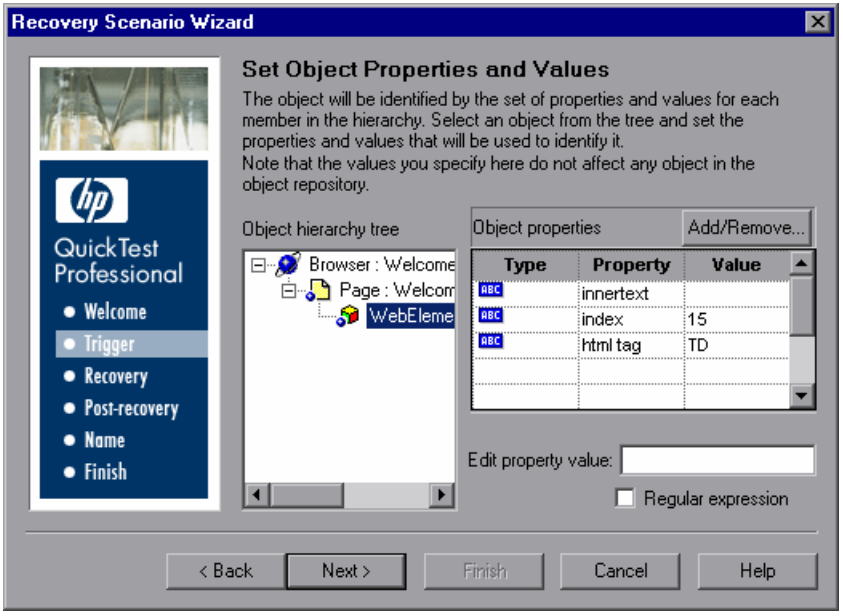
**Note:** The hierarchical object selection tree also enables you to select an object that QuickTest would not ordinarily learn (a non-parent object), such as a Web table.

---

Click **Next** to continue to the Set Object Properties and Values Screen (described on page 788).

## Set Object Properties and Values Screen

After you select the object whose properties you want to specify in the Select Object Screen (described on page 786), the Set Object Properties and Values screen opens.



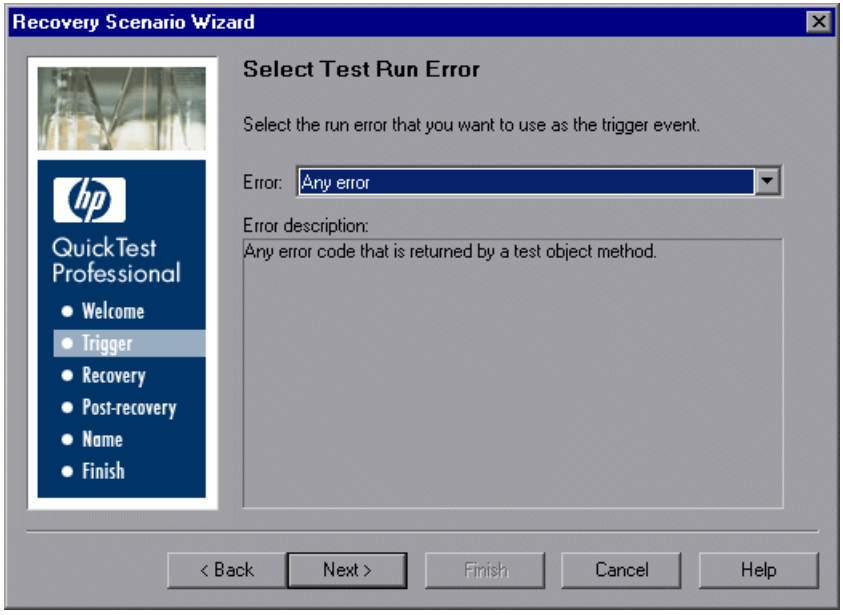

For each object in the hierarchy, in the **Edit property value** box, you can modify the property values used to identify the object. You can also click the **Add/Remove** button to add or remove object properties from the list of property values to check. Note that an object is identified only by its property values, and not by its class.

Select the **Regular expression** check box if you want to use regular expressions in the property value. For information on regular expressions, see the *HP QuickTest Professional User's Guide*.

Click **Next** to continue to the Recovery Operations Screen (described on page 792).

## Select Test Run Error Screen

If you chose a **Test run error** trigger in the Select Trigger Event Screen (described on page 782), the Select Test Run Error screen opens.




In the **Error** list, choose the run error that you want to use as the trigger event:

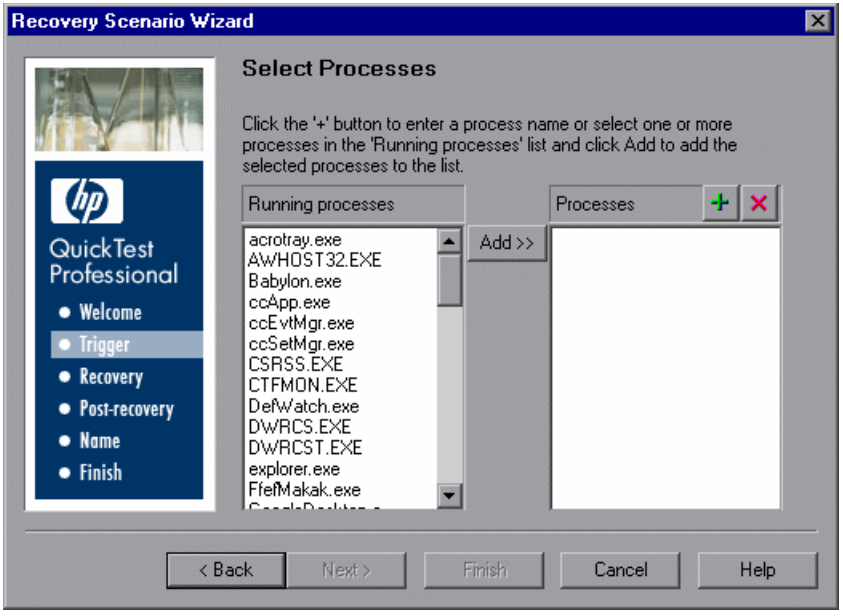
- **Any error.** Any error code that is returned by a test object method.
- **Item in list or menu is not unique.** Occurs when more than one item in the list, menu, or tree has the name specified in the method argument.
- **Item in list or menu not found.** Occurs when QuickTest cannot identify the list, menu, or tree item specified in the method argument. This may be due to the fact that the item is not currently available or that its name has changed.
- **More than one object responds to the physical description.** Occurs when more than one object in your application has the same property values as those specified in the test object description for the object specified in the step.

- **Object is disabled.** Occurs when QuickTest cannot perform the step because the object specified in the step is currently disabled.
- **Object not found.** Occurs when no object within the specified parent object matches the test object description for the object.
- **Object not visible.** Occurs when QuickTest cannot perform the step because the object specified in the step is not currently visible on the screen.

Click **Next** to continue to the "Recovery Operations Screen" on page 792.

## Select Processes Screen

If you chose an **Application crash** trigger in the Select Trigger Event Screen (described on page 782), the Select Processes screen opens.




The **Running processes** list displays all application processes that are currently running. The **Processes** list displays the application processes that will trigger the recovery scenario if they crash.

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.

➤ 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).

➤ To add a process directly to the **Processes** list, click the **Add New Process** button to enter the name of any process you want to add to the list.

➤ To remove a process from the **Processes** list, select it and click the **Remove Process** button.

---

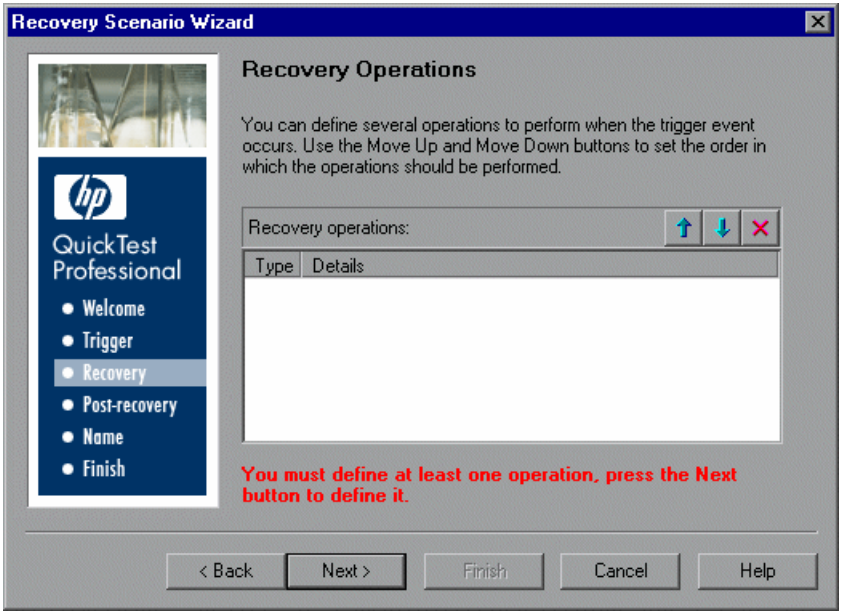
**Tip:** You can modify the name of a process by selecting it in the **Processes** list and clicking the process name to edit it.

---

Click **Next** to continue to the Recovery Operations Screen (described on page 792).

## Recovery Operations Screen

The Recovery Operations screen enables you to manage the collection of recovery operations in the recovery scenario. Recovery operations are operations that QuickTest performs sequentially when it recognizes the trigger event.




You must define at least one recovery operation. To define a recovery operation and add it to the **Recovery operations** list, click **Next** to continue to the Recovery Operation Screen (described on page 793).

If you define two or more recovery operations, you can select a recovery operation and use the **Move Up** or **Move Down** buttons to change the order in which QuickTest performs the recovery operations. You can also select a recovery operation and click the **Remove** button to delete a recovery operation from the recovery scenario.

---

**Note:** 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.

---

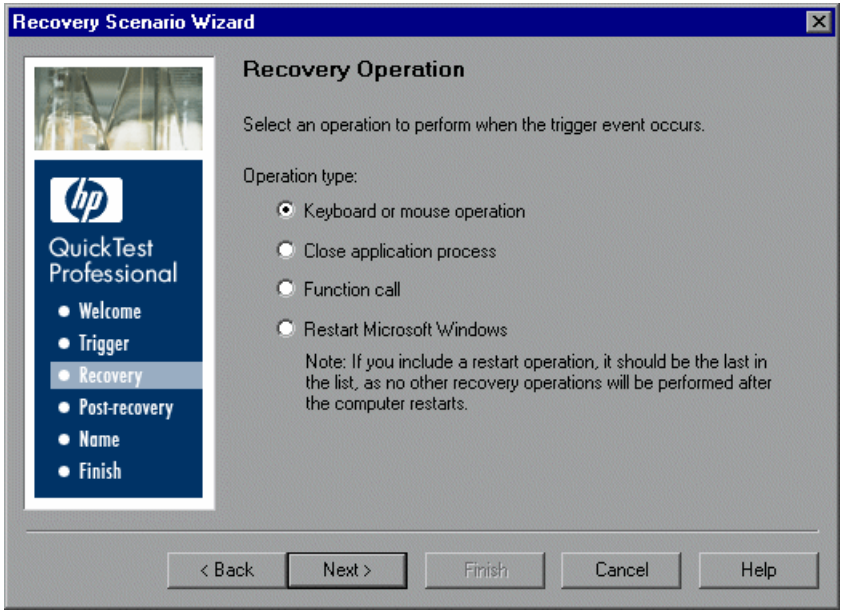
After you have defined at least one recovery operation, the **Add another recovery operation** check box is displayed.

➤ 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 Screen (described on page 801).

## Recovery Operation Screen

The Recovery Operation screen enables you to specify the operations QuickTest performs after it detects the trigger event.




Select a type of recovery operation and click **Next**. The next screen displayed in the wizard depends on which recovery operation type you select.

You can define the following types of recovery operations:

- **Keyboard or mouse operation.** QuickTest simulates a click on a button in a window or a press of a keyboard key. Select this option and click **Next** to continue to the Recovery Operation - Click Button or Press Key Screen (described on page 795).
- **Close application process.** QuickTest closes specified processes. Select this option and click **Next** to continue to the Recovery Operation - Close Processes Screen (described on page 797).
- **Function call.** QuickTest calls a VBScript function. Select this option and click **Next** to continue to the Recovery Operation - Function Call Screen (described on page 798).
- **Restart Microsoft Windows.** QuickTest restarts Microsoft Windows. Select this option and click **Next** to continue to the Recovery Operations Screen (described on page 792).

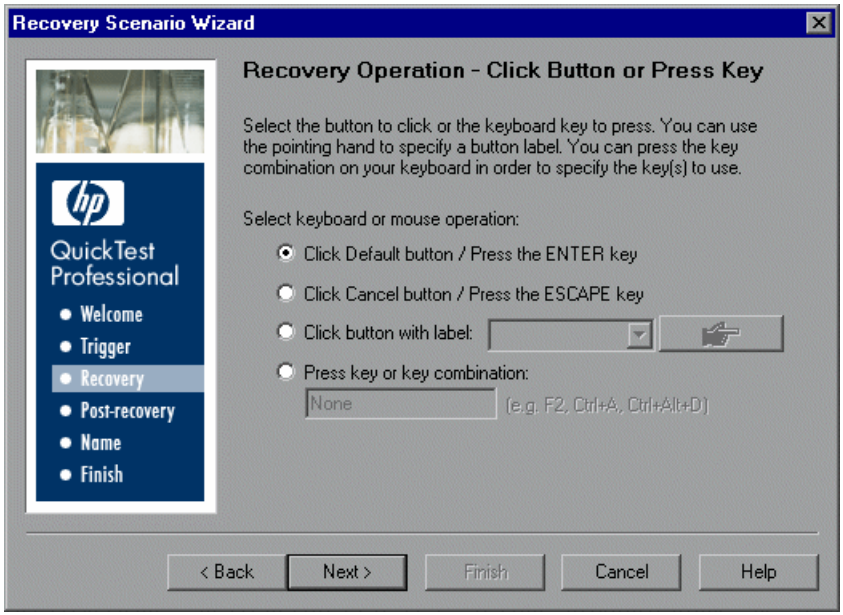
---

**Note:** If you use the **Restart Microsoft Windows** recovery operation, you must ensure that any component associated with this recovery scenario is saved before you run it. You must also configure the computer on which the component is run to automatically log in on restart.

---

## Recovery Operation - Click Button or Press Key Screen

If you chose a **Keyboard or mouse operation** recovery operation in the Recovery Operation Screen (described on page 793), the Recovery Operation – Click Button or Press Key screen opens.




Specify the keyboard or mouse operation that you want QuickTest to perform when it detects the trigger event:

- **Click Default button / Press the ENTER key.** Instructs QuickTest to click the default button or press the ENTER key in the displayed window when the trigger occurs.
- **Click Cancel button / Press the ESCAPE key.** Instructs QuickTest to click the **Cancel** button or press the ESCAPE key in the displayed window when the trigger occurs.

➤ **Click button with label.** Instructs QuickTest to click the button with the specified label in the displayed window when the trigger occurs. If you select this option, click the pointing hand and then click anywhere in the trigger window.

---

**Tip:** Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

---

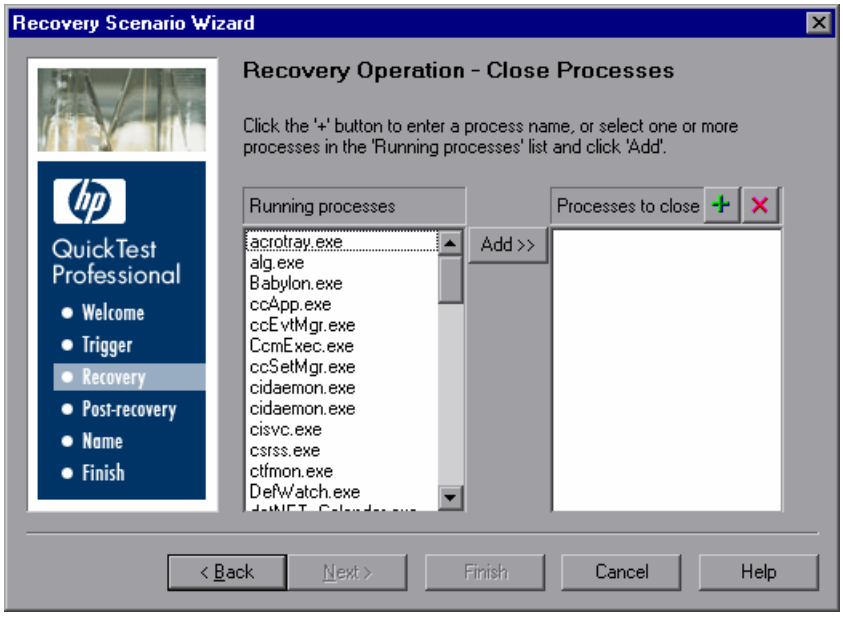
All button labels in the selected window are displayed in the list box. Select the required button from the list.

➤ **Press key or key combination.** Instructs QuickTest to press the specified keyboard key or key combination in the displayed window when the trigger occurs. If you select this option, click in the edit box and then press the key or key combination on your keyboard that you want to specify.

Click **Next**. The Recovery Operations Screen reopens, showing the keyboard or mouse recovery operation that you defined.

## Recovery Operation - Close Processes Screen

If you chose a **Close application process** recovery operation in the Recovery Operation Screen (described on page 793), the Recovery Operation – Close Processes screen opens.




The **Running processes** list displays all application processes that are currently running. The **Processes to close** list displays the application processes that will be closed when the trigger is activated.

- 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).

- To add a process directly to the **Processes to close** list, click the **Add New Process** button to enter the name of any process you want to add to the list.

➤ To remove a process from the **Processes to close** list, select it and click the **Remove Process** button.

---

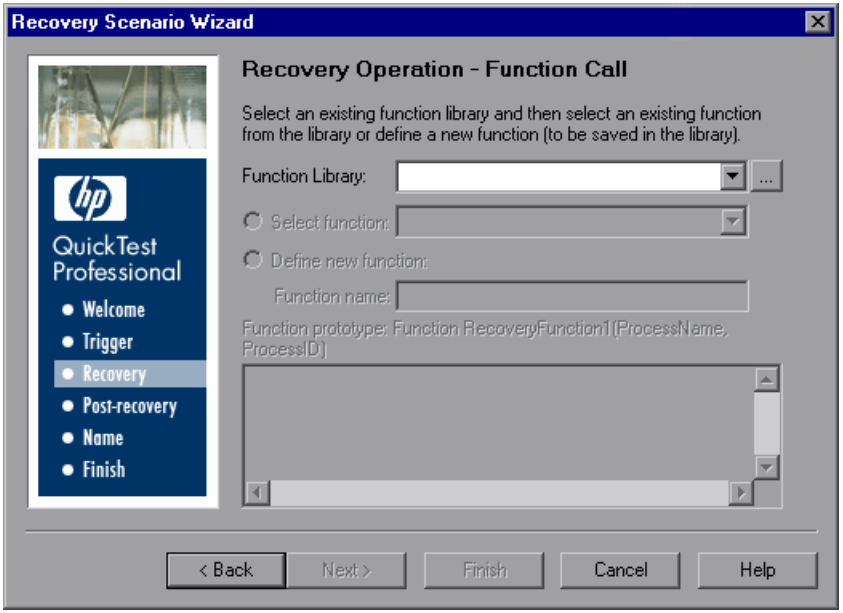
**Tip:** You can modify the name of a process by selecting it in the **Processes to close** list and clicking the process name to edit it.

---

Click **Next**. The Recovery Operations Screen reopens, showing the close processes recovery operation that you defined.

## Recovery Operation - Function Call Screen

If you chose a **Function call** recovery operation in the Recovery Operation Screen (described on page 793), the Recovery Operation – Function Call screen opens.



Select a recently specified function library in the **Function Library** box. Alternatively, click the browse button to navigate to an existing function library.

---

**Note:** The function library must be stored in the Quality Center project.

---

After you select a function library, choose one of the following options:

➤ **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 Screen" on page 782 are displayed.

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.
)
```

➤ **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.

---

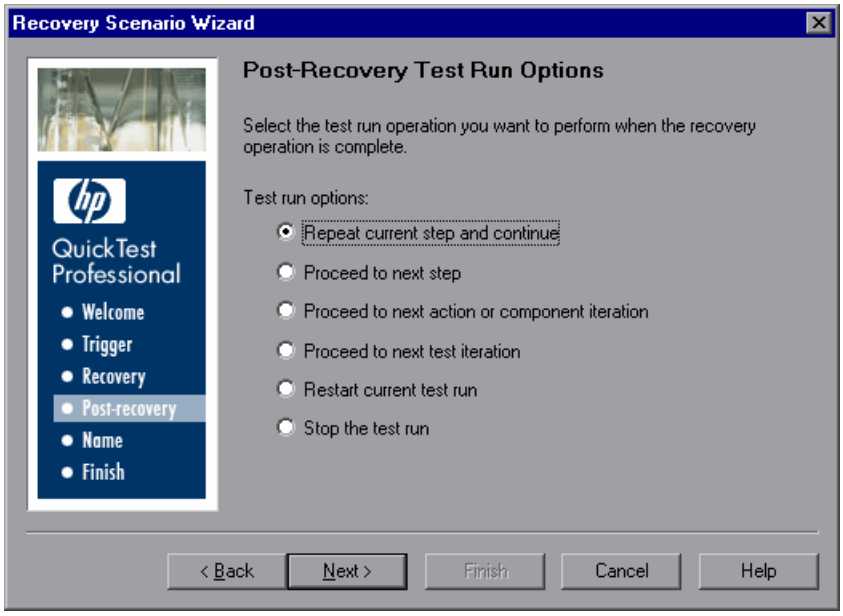
**Note:** If more than one scenario uses a function with the same name from different function libraries, the recovery process may fail. In this case, information regarding the recovery failure is displayed during the run session.

---

Click **Next**. The Recovery Operations Screen (described on page 792) reopens, showing the function operation that you defined.

## Post-Recovery Test Run Options Screen

When you clear the **Add another recovery operation** check box in the Recovery Operations Screen (described on page 792) and click **Next**, the Post-Recovery Test Run Options screen opens. Post-recovery test run options specify how to continue the run session after QuickTest has identified the event and performed all of the specified recovery operations.




QuickTest can perform one of the following run session options after it performs the recovery operations you defined:

➤ **Repeat current step and continue**

The current step is the step that QuickTest was running when the recovery scenario was triggered. If you are using the **On error** activation option for recovery scenarios, the step that returns the error is often one or more steps later than the step that caused the trigger event to occur.

Thus, in most cases, repeating the current step does not repeat the trigger event. For more information, see “Enabling and Disabling Recovery Scenarios” on page 813.

➤ **Proceed to next step**

Skips the step that QuickTest was running when the recovery scenario was triggered. Keep in mind that skipping a step that performs operations on your application may cause subsequent steps to fail.

➤ **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 QuickTest test or business process test iteration from the beginning (or stops running the test if no additional iterations of the test are required).

➤ **Restart current test run**

Stops performing steps and re-runs the component from the beginning.

➤ **Stop the test run**

Stops running the component.

---

**Note:** If you chose **Restart Microsoft Windows** as a recovery operation, you can choose from only the last two test run options listed above.

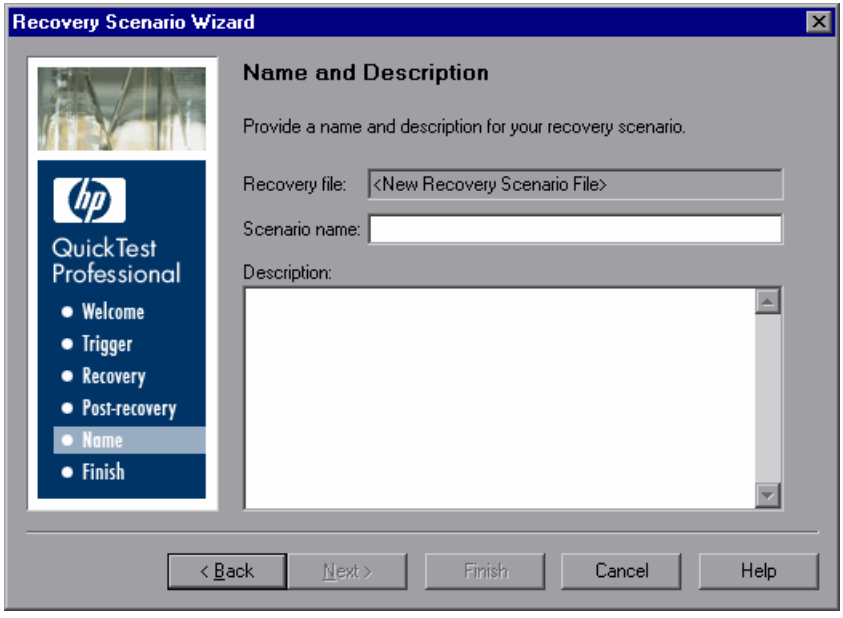
---

Select a test run option and click **Next** to continue to the Name and Description Screen (described on page 803).

## Name and Description Screen

After you specify a test run option in the Post-Recovery Test Run Options Screen (described on page 801), and click **Next**, the Name and Description screen opens.

In the Name and Description screen, you specify a name by which to identify your recovery scenario. You can also add descriptive information regarding the scenario.

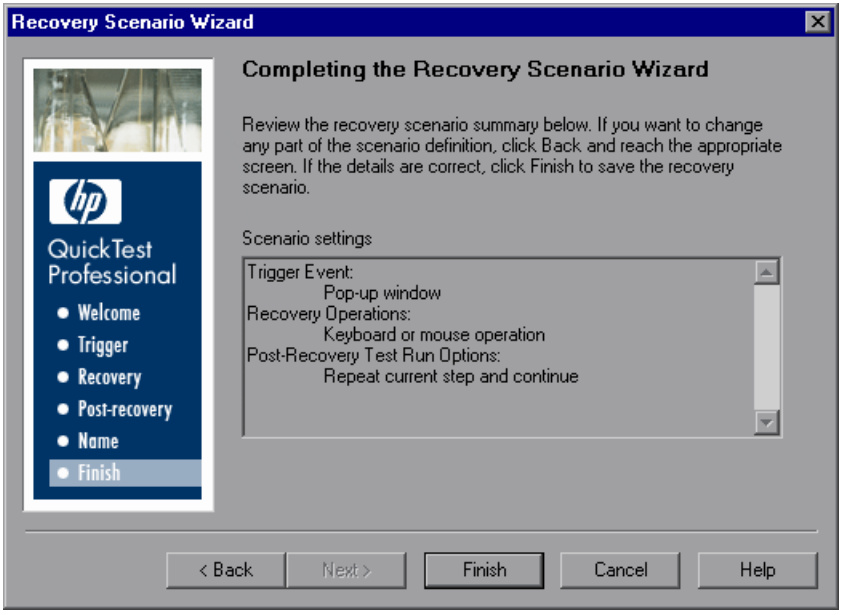



Enter a name and a textual description for your recovery scenario, and click **Next** to continue to the Completing the Recovery Scenario Wizard Screen (described on page 804).

## Completing the Recovery Scenario Wizard Screen

After you specify a recovery scenario name and description in the Name and Description Screen (described on page 803) and click **Next**, the Completing the Recovery Scenario Wizard screen opens.

In the Completing the Recovery Scenario Wizard screen, you can review a summary of the scenario settings you defined.




---

**Note:** 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 more information, see "Working with Application Areas" on page 413.

---

Click **Finish** to complete the recovery scenario definition.

## Saving the Recovery Scenario in a Recovery File

After you create or modify a recovery scenario in a recovery file using the Recovery Scenario Wizard, you need to save the recovery file.

---

**Tip:** If you have not yet saved the recovery file, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes**, and proceed with step 2 below. If you added or modified scenarios in an existing recovery file, and you click **Yes** to the message prompt, the recovery file and its scenarios are saved.

---

**To save a new or modified recovery file:**

1 Click the **Save** button. If you added or modified scenarios in an existing recovery file, the recovery file and its scenarios are saved. If you are using a new recovery file, the Save Attachment dialog box opens.

---

**Tip:** You can also click the arrow to the right of the **Save** button and select **Save As** to save the recovery file under a different name.

---

2 Choose the folder in which you want to save the file.

3 Type a name for the file in the **File name** box and click **Save**.

---

**Note:** When you save a path to a resource, QuickTest checks if the path, or a part of the path, exists in the Folders tab of the Options dialog box (**Tools** > **Options** > **Folders**). 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 tab and define the path relatively.

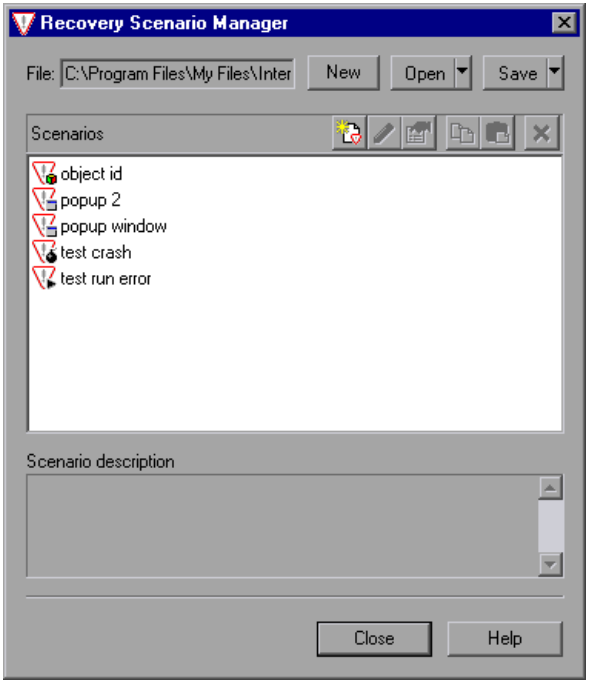
For more information, see “Using Relative Paths in QuickTest” on page 324.

---

The recovery file is saved in the specified location with the file extension **.qrs**.

# Managing Recovery Scenarios

Once you have created recovery scenarios, you can use the Recovery Scenario Manager to manage them.



The Recovery Scenario Manager contains the following recovery scenario 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 component does not run successfully. |
|  | Indicates that the recovery scenario is triggered when an open application fails during the run session. |

The Recovery Scenario Manager enables you to manage existing scenarios by:

- Viewing Recovery Scenario Properties
- Modifying Recovery Scenarios
- Deleting Recovery Scenarios
- Copying Recovery Scenarios between Recovery Scenario Files

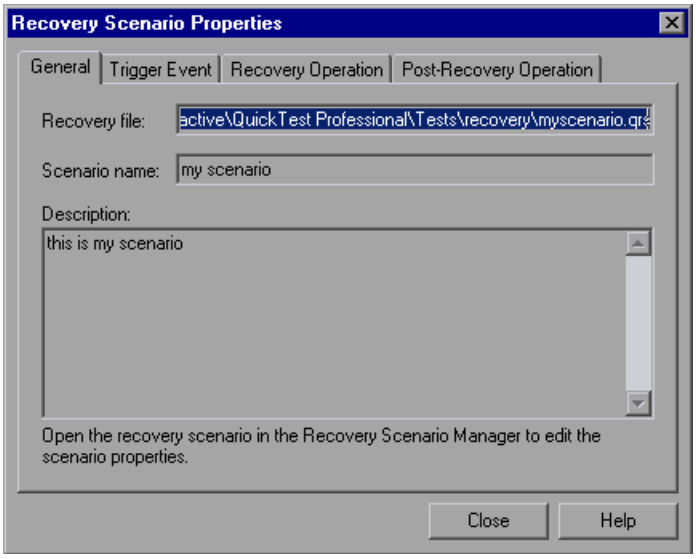
## Viewing Recovery Scenario Properties

You can view properties for any defined recovery scenario.

**To view recovery scenario properties:**

1 In the **Scenarios** box, select the recovery scenario whose properties you want to view.

**2** Click the **Properties** button. Alternatively, you can double-click a scenario in the **Scenarios** box. The Recovery Scenario Properties dialog box opens.




The Recovery Scenario Properties dialog box displays the following read-only information about the selected scenario:

- **General tab.** Displays 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 tab.** Displays the settings for the trigger event defined for the recovery scenario.
- **Recovery Operation tab.** Displays the recovery operations defined for the recovery scenario.
- **Post-Recovery Operation tab.** Displays the post-recovery operation defined for the recovery scenario.

## Modifying Recovery Scenarios

You can modify the settings for an existing recovery scenario.

**To modify a recovery scenario:**

1 In the **Scenarios** box, select the scenario that you want to modify.

2 Click the **Edit** button. The Recovery Scenario Wizard opens, with the settings you defined for the selected recovery scenario.

3 Navigate through the Recovery Scenario Wizard and modify the details as needed. For information on the Recovery Scenario Wizard options, see "Defining Recovery Scenarios" on page 777.

---

**Note:** Modifications you make are not saved until you click **Save** in the Recovery Scenario Manager dialog box. If you have not yet saved your modifications, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes** to save your changes.

---

## Deleting Recovery Scenarios

You can delete an existing recovery scenario if you no longer need it. When you delete a recovery scenario from the Recovery Scenario Manager, the corresponding information is deleted from the recovery scenario file.

---

**Note:** If a deleted recovery scenario is associated with a component, QuickTest ignores it during the run session.

---

**To delete a recovery scenario:**

1 In the **Scenarios** box, select the scenario that you want to delete.

2 Click the **Delete** button. The recovery scenario is deleted from the Recovery Scenario Manager dialog box.

---

**Note:** The scenario is not actually deleted until you click **Save** in the Recovery Scenario Manager dialog box. If you have not yet saved the deletion, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes** to save the recovery scenario file and delete the scenarios.

---

## Copying Recovery Scenarios between Recovery Scenario Files

You can copy recovery scenarios from one recovery scenario file to another.

**To copy a recovery scenario from one recovery scenario file to another:**

1 In the **Scenarios** box, select the recovery scenario that you want to copy.

2 Click the **Copy** button. The scenario is copied to the Clipboard.

3 Click the **Open** button and select the recovery scenario file to which you want to copy the scenario, or click the **New** button to create a new recovery scenario file in which to copy the scenario.

4 Click the **Paste** button. The scenario is copied to the new recovery scenario file.

**Notes:**

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.

Modifications you make are not saved until you click **Save** in the Recovery Scenario Manager dialog box. If you have not yet saved your modifications, and you click the **Close** button in the Recovery Scenario Manager dialog box, QuickTest prompts you to save the recovery file. Click **Yes** to save your changes.

## Associating Recovery Scenarios with Your Application Areas

After you create recovery scenarios, you associate them with selected components (via the application area) so that QuickTest will perform the appropriate scenarios during the run sessions if a trigger event occurs. You can prioritize the scenarios and set the order in which QuickTest applies the scenarios during the run session. You can also choose to disable specific scenarios, or all scenarios, that are associated with an application area.

**Note:** You define recovery scenarios for components in the application area. For more information, see "Working with Application Areas" on page 413.

## Viewing Recovery Scenario Properties

You can view properties for any recovery scenario associated with your application area.

---

**Note:** You modify recovery scenario settings from the Recovery Scenario Manager dialog box. For more information, see "Modifying Recovery Scenarios" on page 809.

---

**To view recovery scenario properties:**

1 In the General pane of the application area, click the **Additional Settings** button. The Application Area Settings dialog box opens.

2 Click the **Recovery** tab.

3 In the **Scenarios** box, select the recovery scenario whose properties you want to view.

4 Click the **Properties** button. Alternatively, you can double-click a scenario in the **Scenarios** box. The Recovery Scenario Properties dialog box opens, displaying read-only information regarding the settings for the selected scenario. For more information, see "Viewing Recovery Scenario Properties" on page 807.

## Setting Recovery Scenario Priorities

You can specify the order in which QuickTest performs associated scenarios during a run session. When a trigger event occurs, QuickTest checks for applicable recovery scenarios in the order in which they are displayed in the Recovery tab of the Application Area Settings dialog box.

**To set recovery scenario priorities:**

1 In the General pane of the application area, click the **Additional Settings** button. The Application Area Settings dialog box opens.

2 Click the **Recovery** tab.

3 In the **Scenarios** box, select the scenario whose priority you want to change.

**4** Click the **Up** or **Down** button. The selected scenario's priority changes according to your selection.

**5** Repeat steps 3-4 for each scenario whose priority you want to change.

## Removing Recovery Scenarios from Your Application Area

You can remove the association between a specific scenario and an application area using the Application Area Settings dialog box. After you remove a scenario from an application area, the scenario itself still exists, but QuickTest will no longer perform the scenario during a run session.

**To remove a recovery scenario from your application area:**

**1** In the General pane of the application area, click the **Additional Settings** button. The Application Area Settings dialog box opens.

**2** Click the **Recovery** tab.

**3** In the **Scenarios** box, select the scenario you want to remove.

**4** Click the **Remove** button. The selected scenario is no longer associated with the application area.

## Enabling and Disabling Recovery Scenarios

You can enable or disable specific scenarios and determine when QuickTest activates the recovery scenario mechanism in the Recovery tab of the Application Area Settings dialog box. When you disable a specific scenario, it remains associated with the application area, but is not performed by QuickTest during the run session. You can enable the scenario at a later time.

You can also specify the conditions for which the recovery scenario is to be activated.

**To enable/disable specific recovery scenarios:**

1. In the General pane of the application area, click the **Additional Settings** button. The Application Area Settings dialog box opens.
2. Click the **Recovery** tab.
3. In the **Scenarios** box, perform one of the following:
   - Select the check box to the left of one or more individual scenarios to enable them.
   - Clear the check box to the left of one or more individual scenarios to disable them.

**To define when the recovery mechanism is activated:**

Select one of the following options in the **Activate recovery scenarios** box:

- **On every step.** The recovery mechanism is activated after every step. Note that choosing **On every step** may result in slower performance during the run session.
- **On error.** The recovery mechanism is activated only after steps that return an error return value.

  Note that the step that returns an error is often not the same as the step that causes the exception event to occur.

  For example, a step that selects a check box may cause a pop-up dialog box to open. Although the pop-up dialog box is defined as a trigger event, QuickTest moves to the next step because it successfully performed the check box selection step. The next several steps could potentially perform checkpoints, functions or other conditional or looping statements that do not require performing operations on your application. It may only be ten statements later that a step instructs QuickTest to perform an operation on the application that it cannot perform due to the pop-up dialog box. In this case, it is this tenth step that returns an error and triggers the recovery mechanism to close the dialog box. After the recovery operation is completed, the current step is this tenth step, and not the step that caused the trigger event.

➤ **Never.** The recovery mechanism is disabled.

---

**Tip:** You can also enable or disable specific scenarios or all scenarios associated with an application area programmatically during the run session. For more information, see "Programmatically Controlling the Recovery Mechanism" on page 815.

---

### Setting Default Recovery Scenario Settings for All New Components

You define the default recovery scenarios for all new components in the component's application area. For more information, see "Working with Application Areas" on page 413.

## Programmatically Controlling the Recovery Mechanism

You can use the Recovery object to control the recovery mechanism programmatically during the run session. For example, you can enable or disable the entire recovery mechanism or specific recovery scenarios for certain parts of a run session, retrieve status information about specific recovery scenarios, and explicitly activate the recovery mechanism at a certain point in the run session.

By default, QuickTest checks for recovery triggers when an error is returned during the run session. You can use the Recovery object's Activate method to force QuickTest to check for triggers after a specific step in the run session. For example, suppose you know that an object property checkpoint will fail if certain processes are open when the checkpoint is performed. You want to be sure that the pass or fail of the checkpoint is not affected by these open processes, which may indicate a different problem with your application.

However, a failed checkpoint does not result in a run error. So by default, the recovery mechanism would not be activated by the object state. You can define a recovery scenario that looks for and closes specified open processes when an object's properties have a certain state. This state shows the object's property values as they would be if the problematic processes were open.

You can instruct QuickTest to activate the recovery mechanism if the checkpoint fails so that QuickTest will check for and close any problematic open processes and then try to perform the checkpoint again. This ensures that when the checkpoint is performed the second time it is not affected by the open processes.

For more information on the Recovery object and its methods, see the *HP QuickTest Professional Object Model Reference*.

# 33

# Automating QuickTest Operations

Just as you use QuickTest to automate the testing of your applications, you can use the QuickTest Professional automation object model to automate your QuickTest operations. Using the objects, methods, and properties exposed by the QuickTest automation object model, you can write scripts that configure QuickTest options and run components instead of performing these operations manually using the QuickTest interface.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple components, or quickly configuring QuickTest according to your needs for a particular environment or application.

**This chapter includes:**

# About Automating QuickTest Operations

You can use the QuickTest Professional automation object model to write scripts that automate your QuickTest operations. The QuickTest automation object model provides objects, methods, and properties that enable you to control QuickTest from another application.

## 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.

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.

## What is the QuickTest Automation Object Model?

Essentially all configuration and run functionality provided via the QuickTest interface is in some way represented in the QuickTest automation object model via objects, methods, and properties. Although a one-on-one comparison cannot always be made, most dialog boxes in QuickTest have a corresponding automation object, most options in dialog boxes can be set and/or retrieved using the corresponding object property, and most menu commands and other operations have corresponding automation methods.

You can use the objects, methods, and properties exposed by the QuickTest automation object model, along with standard programming elements such as loops and conditional statements to design your script.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple components, or quickly configuring QuickTest according to your needs for a particular environment or application.

For example, you can create and run an automation script from Microsoft Visual Basic that loads the required add-ins for a component, starts QuickTest in visible mode, opens the component, configures settings that correspond to those in the Options, Business Component Settings, and Record and Run Settings dialog boxes, runs the component, and saves the component.

You can then add a simple loop to your script so that your single script can perform the operations described above for multiple components.

You can also create an initialization script that opens QuickTest with specific configuration settings. You can then instruct all of your testers to open QuickTest using this automation script to ensure that all of your testers are always working with the same configuration.

## Deciding When to Use QuickTest Automation Scripts

Creating a useful QuickTest automation script requires planning, design time, and testing. You must always weigh the initial investment with the time and human-resource savings you gain from automating potentially long or tedious tasks.

Any QuickTest operation that you must perform many times in a row or must perform on a regular basis is a good candidate for a QuickTest automation script.

The following are just a few examples of useful QuickTest automation scripts:

- **Initialization scripts.** You can write a script that automatically starts QuickTest and configures the options and the settings required for testing a specific environment.

- **Maintaining your components.** You can write a script that iterates over your collection of components to accomplish a certain goal. For example:
  - **Updating values.** You can write a script that opens each 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 components to match the updated values in your application.
  - **Applying new options to existing components.** When you upgrade to a new version of QuickTest, you may find that the new version offers certain options that you want to apply to your existing components. You can write a script that opens each existing component, sets values for the new options, then saves and closes it.
- **Calling QuickTest from other applications.** You can design your own applications with options or controls that run QuickTest automation scripts. For example, you could create a Web form or simple Windows interface from which a product manager could schedule QuickTest runs, even if the manager is not familiar with QuickTest.

## Choosing a Language and Development Environment for Designing and Running Automation Scripts

You can choose from a number of object-oriented programming languages for your automation scripts. For each language, there are a number of development environments available for designing and running your automation scripts.

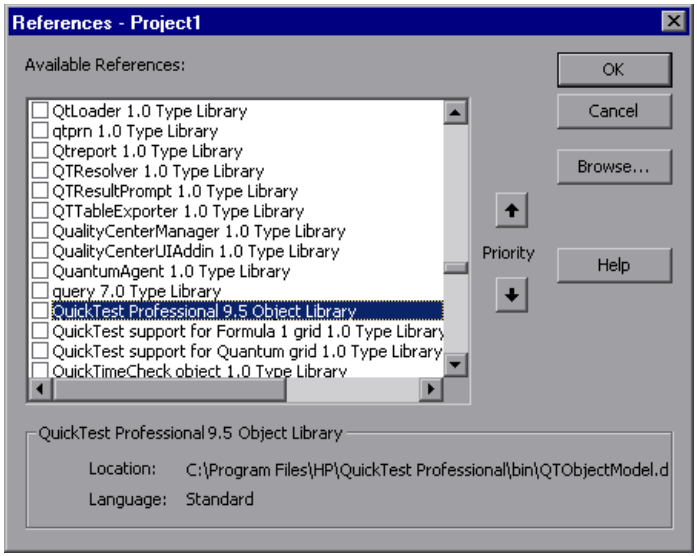
### Writing Your Automation Script

You can write your QuickTest automation scripts in any language and development environment that supports automation. For example, you can use: VBScript, JavaScript, Visual Basic, Visual C++, or Visual Studio .NET.

Some development environments support referencing a type library. A **type library** is a binary file containing the description of the objects, interfaces, and other definitions of an object model.

If you choose a development environment that supports referencing a type library, you can take advantage of features like Microsoft IntelliSense, automatic statement completion, and status bar help tips while writing your script. The QuickTest automation object model supplies a type library file named **QTObjectModel.dll**. This file is stored in **<QuickTest installation folder>\bin**.

If you choose an environment that supports it, be sure to reference the QuickTest type library before you begin writing or running your automation script. For example, if you are working in Microsoft Visual Basic, choose **Project** > **References** to open the References dialog box for your project. Then select **QuickTest Professional <Version> Object Library** (where <**Version**> is the current installed version of the QuickTest automation type library).



### Running 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
```

## Learning the Basic Elements of a QuickTest Automation Script

Like most automation object models, the root object of the QuickTest automation object model is the **Application** object. The Application object represents the application level of QuickTest. You can use this object to return other elements of QuickTest such as the Test object (which represents a component document), Options object (which represents the Options dialog box), or Addins collection (which represents a set of add-ins from the Add-in Manager dialog box), and to perform operations like loading add-ins, starting QuickTest, opening and saving components, and closing QuickTest.

Each object returned by the Application object can return other objects, perform operations related to the object and retrieve and/or set properties associated with that object.

Every automation script begins with the creation of the QuickTest Application object. Creating this object does not start QuickTest. It simply provides an object from which you can access all other objects, methods and properties of the QuickTest automation object model.

---

**Note:** You can also optionally specify a remote QuickTest computer on which to create the object (the computer on which to run the script). For more information, see the **Running Automation Programs on a Remote Computer** section of the online *QuickTest Automation Object Model Reference*.

---

The structure for the rest of your script depends on the goals of the script. You may perform a few operations before you start QuickTest such as retrieving the associated add-ins for a component, loading add-ins, and instructing QuickTest to open in visible mode.

After you perform these preparatory steps, if QuickTest is not already open on the computer, you can open QuickTest using the Application.Launch method. Most operations in your automation script are performed after the Launch method.

For information on the operations you can perform in an automation program, see the online *HP QuickTest Professional Object Model Reference*. For more information on this Help file, see "Using the QuickTest Automation Reference" on page 824.

When you finish performing the necessary operations, or you want to perform operations that require closing and restarting QuickTest, such as changing the set of loaded add-ins, use the Application.Quit method.

## Generating Automation Scripts

The General tab of the Options dialog box and the Object Identification dialog box each contain a **Generate Script** button. Clicking this button generates an automation script file (**.vbs**) containing the current settings from the corresponding dialog box.

You can run the generated script as is to open QuickTest with the exact configuration of the QuickTest application that generated the script, or you can copy and paste selected lines from the generated files into your own automation script.

For example, the generated script for the Options dialog box may look something like this:

```
Dim App 'As Application
Set App = CreateObject("QuickTest.Application")
App.Launch
App.Visible = True
App.Options.DisableVORecognition = False
App.Options.AutoGenerateWith = False
App.Options.WithGenerationLevel = 2
App.Options.TimeToActivateWinAfterPoint = 500
...
...
App.Options.WindowsApps.NonUniqueListItemRecordMode = "ByName"
App.Options.WindowsApps.RecordOwnerDrawnButtonAs = "PushButtons"
App.Folders.RemoveAll
```

For more information on the **Generate Script** button and for information on the options available in the Options and Object Identification dialog boxes, see Chapter 5, "Configuring Object Identification" and Chapter 20, "Setting Global Testing Options".

## Using the QuickTest Automation Reference

The QuickTest Automation Reference is a Help file that provides detailed descriptions, syntax information, and examples for the objects, methods, and properties in the QuickTest automation object model.

You can open the *QuickTest Automation Reference* from:

- QuickTest program folder (**Start** > **Programs** > **QuickTest Professional** > **Documentation** > **QuickTest Automation Reference**)
- Main QuickTest Help (**Help** > **QuickTest Professional Help** > **QuickTest Advanced References** > **QuickTest Automation**)

# Part X

## Appendix

# A

# Frequently Asked Questions

This chapter answers some of the questions that are asked most frequently by advanced users of QuickTest. The questions and answers are divided into the following sections:

**This chapter includes:**



## Creating Components

- **How can I record on objects or environments not supported by QuickTest?**

  You can do this in a number of ways:

  - Install and load any of the add-ins that are available for QuickTest Professional. QuickTest supports many developmental environments including Java, Oracle, .NET, SAP Solutions, Siebel, PeopleSoft, terminal emulators, and Web services.
  - You can map objects of an unidentified or custom class to standard Windows classes. For more information on object mapping, see "Mapping User-Defined Test Object Classes" on page 206.

➤ QuickTest provides add-in extensibility that you can use to extend QuickTest built-in support for various objects. This enables you to direct QuickTest to recognize an object as belonging to a specific test object class, and to specify the behavior of the test object. You can also extend the list of available test object classes that QuickTest recognizes. This enables you to create components that fully support the specific behavior of your custom objects.

➤ **How can I open an application from a component?**

To add a step that opens an application, select **Operation** from the **Item** column, select **OpenApp** from the **Operation** column, and then enter the the full path in the **Value** column, for example:

C:\Program Files\HP\QuickTest Professional\samples\flight\app\flight4a.exe

➤ **How does QuickTest capture user processes in Web pages?**

QuickTest hooks the Microsoft Internet Explorer browser. As the user navigates the Web-based application, QuickTest records the user actions. (For information on modifying which user actions are recorded, see the section on configuring Web event recording in the *HP QuickTest Professional Add-ins Guide*.) QuickTest can then run the component by running the steps as they originally occurred.

## Working with Function Libraries

➤ **Can I store functions and subroutines in a function library?**

You can create one or more VBScript function libraries containing your functions, and then use them in any component by associating them with the component's associated application area. You can use the QuickTest function library editor to create and debug your function libraries.

You can register your functions as methods for QuickTest test objects. Your registered methods can override the functionality of an existing test object method for the duration of a run session, or you can register a new method for a test object class.

For more information, see Chapter 11, "Working with User-Defined Functions and Function Libraries" and Chapter 12, "Working with Application Areas."

➤ **How can I enter information during a run session?**

You can insert the VBScript InputBox function into an associated function library to create a user-defined function that enables you to display a dialog box that prompts the user for input and then continues running the component. You can use the value that was entered by the user later in the run session. For more information on the InputBox function, see the *VBScript Reference*.

➤ **How do I customize the Test Results?**

You can send messages to the run results report by using the **ReportEvent** method, for example:

```
Reporter.ReportEvent 1, "Custom Step", "The user-defined step failed"
```

The results of each QuickTest run session are saved in a single **.xml** file (called **results.xml**). You can modify this file, as needed. You can use the **QuickTest Test Results Schema** (available from the QuickTest Professional Help) to help you customize your test results.

## Working with Dynamic Content

➤ **How can I create and run 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 QuickTest will recognize them when it runs the component using regular expressions, the **Description** object, repository parameters, or **SetTOProperty** steps.

➤ **How can I check that a child window exists (or does not exist)?**

Sometimes a link in one window creates another window.

You can use the **Exist** property to check whether or not a window exists. For example:

```
If Window("Main").ActiveX("Slider").Exist Then
. . .
```

You can also use the ChildObjects method to retrieve all child objects (or the subset of child objects that match a certain description) on the Desktop or within any other parent object.

Example:

```
Set oDesc = Description.Create
oDesc("Class Name").Value = "Window"

ser coll = Desktop.ChildObjects(oDesc)
For i = 0 to coll.count -1
        msgbox coll(i).GetROProperty("text")
Next
```

For more information on the Exist property and ChildObjects method, see the *HP QuickTest Professional Object Model Reference*.

➤ **How does QuickTest record on dynamically generated URLs and Web pages?**

QuickTest actually clicks links as they are displayed on the page. Therefore, QuickTest records how to find a particular object, such as a link on the page, rather than the object itself. For example, if the link to a dynamically generated URL is an image, then QuickTest records the "IMG" HTML tag, and the name of the image. This enables QuickTest to find this image in the future and click on it.

➤ **How does QuickTest handle tabs in browsers?**

QuickTest provides several methods that you can use with the **Browser** test object to manage tabs in your Web browser.

**OpenNewTab** opens a new tab in the current Web browser.

**IsSiblingTab** indicates whether a specified tab is a sibling of the current tab object in the same browser window.

**Close** closes the current tab if more than one tab exists, and closes the browser window if the browser contains only one tab.

**CloseAllTabs** closes all tabs in a browser and closes the browser window.

For more information on these **Browser**-related methods, see the **Web** section of the *HP QuickTest Professional Object Model Reference*.

# Advanced Web Issues

➤ **How does QuickTest handle cookies?**

Server side connections, such as CGI scripts, can use cookies both to store and retrieve information on the client side of the connection.

QuickTest stores cookies in the memory for each user, and the browser handles them as it normally would.

➤ **Where can I find a Web page's cookie?**

The cookie used by the Internet Explorer browser can be accessed through the browser's Document Object Model (DOM) using the **.Object** property in a function. In the following example the cookie collection is returned the from the browser.

```
Browser("Flight reservations").Page("Flight reservations").Object.Cookie
```

➤ **How does QuickTest handle session IDs?**

The server, not the browser, handles session IDs, usually by a cookie or by embedding the session ID in all links. This does not affect QuickTest.

➤ **How does QuickTest handle server redirections?**

When the server redirects the client, the client generally does not notice the redirection, and misdirections generally do not occur. In most cases, the client is redirected to another script on the server. This additional script produces the HTML code for the subsequent page to be viewed. This has no effect on QuickTest or the browser.

➤ **How does QuickTest handle meta tags?**

Meta tags do not affect how the page is displayed. Generally, they contain information only about who created the page, how often it is updated, what the page is about, and which keywords represent the page's content. Therefore, QuickTest has no problem handling meta tags.

➤ **Does QuickTest work with .asp and .jsp?**

Dynamically created Web pages utilizing Active Server Page technology have an .asp extension. Dynamically created Web pages utilizing Java Server Page technology have a .jsp extension. These technologies are completely server-side and have no bearing on QuickTest.

➤ **Does QuickTest work with COM?**

QuickTest complies with the COM standard.

QuickTest supports COM objects embedded in Web pages (which are currently accessible only using Microsoft Internet Explorer) and you can drive COM objects in VBScript.

➤ **Does QuickTest work with XML?**

XML is eXtensible Markup Language, a pared-down version of SGML for Web documents, that enables Web designers to create their own customized tags. QuickTest supports XML and recognizes XML tags as objects.

For more information, see the *HP QuickTest Professional User's Guide*, and the **XMLUtil** object in the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

➤ **How can I access HTML tags directly?**

QuickTest provides direct access to the Internet Explorer's Document Object Model (DOM) through which you can access the HTML tags directly. Access to the DOM is performed using the .Object notation.

The 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 Test 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 test results.
    Reporter.ReportEvent 0, TagName, InnerText
Next
```

➤ **Where can I find information on the Internet Explorer Document Object Model?**

For information on the Internet Explorer DOM, browse to the following Web sites:

Document object:
http://msdn2.microsoft.com/en-us/library/ms531073.aspx

Other DHTML objects:
http://msdn2.microsoft.com/en-us/library/ms533054.aspx

General DHTML reference:
http://msdn2.microsoft.com/en-us/library/ms533050.aspx

➤ **How can I send keyboard key commands (such as shortcut commands) to objects that do not support the Type method?**

For objects that do not support the **Type** method, use the Windows Scripting **SendKeys** method. For more information, see the Microsoft VBScript Language Reference (choose **Help** > **QuickTest Professional Help** > **VBScript** > **Windows Script Host**).

## Standard Windows Environment

➤ **How can I record on nonstandard menus?**

You can modify how QuickTest behaves when it records menus. The options that control this behavior are located in the Advanced Windows Applications Options dialog box.
(**Tools** > **Options** > **Windows Applications** > **Advanced**).

For more information, see the *HP QuickTest Professional Add-ins Guide*.

# Component Maintenance

➤ **How do I maintain my component when my application changes?**

The way to maintain a 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 components rather than one large component for your entire application.

If you have many components 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 to change the set of test object properties used to identify the objects in your application. For more information, see "Updating a Component Using the Update Run Mode Option" on page 720.

If there is a discrepancy between the test object 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 component in Maintenance Run Mode, QuickTest runs your 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 information, see "Running Components with the Maintenance Run Wizard" on page 704.

➤ **How can I remove test result files from old components?**

You can use the Test Results Deletion Tool to view a list of all of the test results in a specific location in your file system or in your Quality Center project. You can then delete any test results that you no longer require.

The Test Results Deletion Tool enables you to sort the test results by name, date, size, and so forth, so that you can more easily identify the results you want to delete.

To open this utility, choose **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Test Results Deletion Tool**.

# Improving QuickTest Performance

You can improve the working speed of QuickTest by doing any of the following:

- In the Add-in Manager, load only the add-ins you need for a specific QuickTest session when QuickTest starts. This will improve performance while learning objects and during run sessions. For more information on loading add-ins, see the *HP QuickTest Professional Add-ins Guide*.
- Try to use the same application area for all components in a business process test, as this improves performance.
- Run your components in "fast mode." From the Run tab in the Options dialog box, select the **Fast** option. This instructs QuickTest to run your component without displaying the execution arrow for each step, enabling the component to run faster. For more information on the Run tab of the Options dialog box, see “Setting Run Testing Options” on page 591.
- Decide when you want to capture and save images and/or movies of the application for the run session results. From the Run tab in the Options dialog box, select an option from the **Save still image captures to results** box and/or select the **Save movie to results** check box and specify the required settings. You can improve run time and reduce disk space by saving screen captures and movie segments only in certain situations, such as when errors occur, or by not saving them at all. For more information on the Run tab of the Options dialog box, see “Setting Run Testing Options” on page 591.
- Save the test results report to a temporary folder to overwrite the results from the previous run session every time you run a component. For more information, see “Running Your Entire Component” on page 618.
- Use the Results Deletion Tool to remove unwanted or obsolete run results from your system, according to specific criteria that you define. This enables you to free up valuable disk space. For more information, see “Deleting Results Using the Test Results Deletion Tool” on page 651. You can also use the

# B

# Creating Custom Process Guidance Packages

This chapter guides you through the process of creating custom process guidance packages. You can distribute your custom packages to the QuickTest users in your organization. QuickTest users can then display the processes from your package in QuickTest while they work, to assist them in following your organization's processes and standards.

**This chapter includes:**



## About Process Guidance Packages

A Process Guidance Package is comprised of two entities: the package configuration file and the data files.

- **Package Configuration file**. This XML file defines the **Processes** included in the package and the structure of the **Groups** and **Activities** in each process.
- **Data Files**. A set of HTML files. Each HTML file contains the content for a single activity.

# Understanding the Package Configuration File

To create a new package, you first create an XML file that describes the processes included in the package and sets the structure of the groups and activities in each process. This structure is displayed as a table of contents for a selected process in the QuickTest **Process Guidance Activities** pane.

---

**Important:** Save the configuration file with the name: **Configuration.xml**

---

The following is an example of a package configuration file that contains two processes:

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessGuidance Name="MyCustomPackage">
   <Process Name="My Process" ID="Process1" DocType="test" Addin="web"
SortLevel="4" >
      <Group Name="New User Overview">
         <Activity Name="Step 1" Address="Step1.html"  />
         <Activity Name="Step 2" Address="Step2.html"  />
      </Group>
   </Process>
   <Process Name="Important Processes" ID="Process2" DocType="test|AA"
SortLevel="3">
      <Group Name="Getting Started">
         <Activity Name="Open" Address="F:\ProcessData\open.html"  />
         <Activity Name="Create" Address="F:\ProcessData\create.html"  />
         <Activity Name="Test" Address="F:\ProcessData\test.html"  />
         <Activity Name="Debug" Address="F:\ProcessData\debug.html"  />
      </Group>
      <Group Name="Finish">
         <Activity Name="Save" Address="F:\ProcessData\save.html"  />
         <Activity Name="Close" Address="F:\ProcessData\close.html"  />
         <Activity Name="Exit" Address="F:\ProcessData\exit.html"  />
      </Group>
   </Process>
</ProcessGuidance>
```

## XML Details

The elements and attributes you can use in your package configuration file are described in this section.

- **<Process> Element.** Defines a new process. This element supports the following attributes:
  - **Name**. The name of the process as you want it to appear in the QuickTest Process Guidance pane.
  - **ID**. A unique identification name. This name is used to distinguish between two processes with the same name.
  - **DocType**. Indicates the QuickTest document types for which this process is applicable. If specified, the process is available only when the relevant document type is open.

    In the example above, if a QuickTest user opens a test document, both processes will be available, but if an application area document is opened, only the second process will be available.

    **Possible values**:
    - **test**. A test document.
    - **AA**. An application area document.
    - **BC**. A business component document.
    - **SBC**. A scripted component document.
  - **Addin**. Indicates the QuickTest add-ins for which this process is applicable. If specified, the process is available only when the relevant add-in is loaded.

    In the example above, the first process will be available only if the Web Add-in is loaded. The second process will always be visible.

    Specify the add-in value using the add-in name as displayed in the Add-in Manager.
  - **SortLevel**. Determines the location of the process within the process list. This list is displayed in the Process Guidance Management dialog box and in the QuickTest **Automation** > **Process Guidance List** menu.

- **<Group> Element**. Defines a new group in the process. This element supports the following attributes:
  - **Name**. Same as the **Name** attribute for the **<Process>** element, as described above.
  - **ID**. Same as the **ID** attribute for the **<Process>** element, as described above.
  - **Addin**. Same as the **Addin** attribute for the **<Process>** element, as described above.
- **<Activity> Element.** Defines an activity within the group.
  - **Name**. Same as the **Name** attribute for the **<Process>** element, as described above.
  - **ID**. Same as the **ID** attribute for the **<Process>** element, as described above.
  - **Addin**. Same as the **Addin** attribute for the **<Process>** element, as described above.
  - **Address**. The path where the relevant HTML data file is located. This can be a local or network path on the file system or an HTTP address. If you specify a relative path, the location is resolved relative to the configuration file location.

# Creating Data Files

Each data file contains the HTML content for a single process guidance activity. When an activity link is clicked in the **Process Guidance Activities** pane, the HTML content is displayed in a browser control in the **QuickTest Process Guidance Description** pane.

The package data files can include reference to a **.css** file to display content in your organization's standard style, and can contain any content that can be displayed by a browser.

The HTML files, and any folders or files that the HTML files reference can be stored on the user's local hard drive in a network location on the file system or on a Web server. The package configuration file (the **Address** attribute of each **Activity** element) provides HTML links for each activity.

It is recommended that the HTML file for each activity be written such that there will be minimum scrolling when the content is displayed in the Process Guidance Description pane at its default size.

If you find that your HTML files are too long, you may want to break them up into multiple process guidance activities to make it easier for your QuickTest users to reference while they work.

# Installing Custom Process Guidance Packages in QuickTest

There are two ways to distribute and install custom process guidance packages:

- Install the process guidance package from a zip file
- Install the process guidance package via registry key

### Install the process guidance package from a zip file

1. Create a folder that contains the **Configuration.xml** file and all the HTML data files (as well as any files or folders referenced from the HTML files).
2. Zip the folder and then send the **.zip** file to all relevant QuickTest users or store it in a location that they can access.
3. In QuickTest, choose **File** > **Process Guidance Management**. The Process Guidance Management dialog box opens.
4. Click the **Add** button and browse to the **.zip** file. The package is added and its processes are displayed in the dialog box.

### Install the process guidance package via registry key

1. Prepare the **Configuration.xml** file and the data files.
2. Place the data files in a local or shared network folder or on a Web server. Ensure that the **Address** attribute of the **Activity** elements in the **Configuration.xml** file point to this location.
3. Copy the **Configuration.xml** to a local drive on the QuickTest computer.
4. Open the Registry Editor and find the key:

   **HKEY_LOCAL_MACHINE\SOFTWARE\Mercury Interactive\QuickTest Professional\MicTest\ProcessGuidance\ConfFiles**
5. Add a value to this key with the path to the **Configuration.xml** file. The next time QuickTest is opened, it will include the new package.

# Index

## A

## D

## E



## F

## G



## H



## I



## J



## K

## L



## M

## N



## O

## P

## Q

## S

## T

## U



## V



## W

## X