OPTIMIZE

MERCURY QUICKTEST PROFESSIONAL™

FOR BUSINESS PROCESS TESTING

VERSION 9.1

User's Guide

MERCURY™

BUSINESS TECHNOLOGY OPTIMIZATION

Mercury QuickTest Professional for Business Process Testing

User's Guide

Version 9.1

Document Release Date: October 1, 2006

MERCURY™

Mercury QuickTest Professional for Business Process Testing User's Guide, Version 9.1

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

Mercury Interactive Corporation
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Toll Free: (800) TEST-911
Customer Support: (877) TEST-HLP
Fax: (650) 603-5300

If you have any comments or suggestions regarding this document, please send them by e-mail to documentation@mercury.com.

QTP4BPTUG9.1/01

Table of Contents

PART I: INTRODUCING BUSINESS PROCESS TESTING

PART II: WORKING WITH APPLICATION AREAS AND COMPONENTS

PART III: CONFIGURING SETTINGS

PART IV: WORKING WITH TEST OBJECTS AND OBJECT REPOSITORIES

PART V: RUNNING AND DEBUGGING COMPONENTS

PART VI: WORKING WITH ADVANCED FEATURES

PART VII: 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 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.

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

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 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, how to handle missing resources, and how to load and work with QuickTest built-in add-ins.

Part III Configuring Settings

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

Part IV Working with Test Objects and Object Repositories

Describes how QuickTest identifies objects in your application and how to work with object repositories, including how to manage and merge them.

Part V Running and Debugging Components

Describes how to run components and their associated function libraries, analyze results, and control run sessions to identify and isolate bugs.

Part VI Working with Advanced Features

Describes how to work with advanced QuickTest features, including working with recovery scenarios, and creating and working with user-defined functions and function libraries. This part also describes how to automate QuickTest operations.

Part VII Appendixes

Provides information on frequently asked questions about QuickTest.

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 *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** > **Mercury QuickTest Professional Installation Guide**.

QuickTest Professional Tutorial teaches you basic QuickTest skills and shows you how to design tests for your applications. Choose **Help** > **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** 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 Object Model** 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 XML schema that enables you to customize your test results.
  - **QuickTest Test Object Schema** documents the XML schema that enables you to extend test object support in different environments.
  - **QuickTest Object Repository Automation** documents the Object Repository automation object model that enables you 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.

Choose **Help** > **QuickTest Professional Help**. Online Help is also available from specific QuickTest windows and dialog boxes by clicking in the window and pressing F1. You can also view a description, syntax, and examples for a QuickTest test object, method, or property by placing the cursor on it and pressing F1.

Note: Your QuickTest Help may contain additional items relevant to any QuickTest add-ins you have installed. For more information, refer to the relevant add-in documentation.

Additional Online Resources

Mercury Tours sample Web site is the basis for many examples in this guide. The URL for this Web site is http://newtours.mercury.com. Choose **Start** > **Programs** > **QuickTest Professional** > **Sample Applications** > **Mercury Tours Web Site**.

Knowledge Base uses your default Web browser to open the Mercury Customer Support Web Site directly to the Knowledge Base landing page. Choose **Help** > **Knowledge Base**. The URL for this Web site is http://support.mercury.com/cgi-bin/portal/CSO/kbBrowse.jsp.

Customer Support Web Site uses your default Web browser to open the Mercury Customer Support Web site. This site enables you to browse the Mercury 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 http://support.mercury.com.

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

Mercury Home Page uses your default Web browser to access Mercury's Web site. This site provides you with the most up-to-date information on Mercury and its products. This includes new software releases, seminars and trade shows, customer support, educational services, and more. Choose **Help** > **Mercury Home Page**. The URL for this Web site is http://www.mercury.com.

Mercury Best Practices contain guidelines for planning, creating, deploying, and managing a world-class IT environment. Mercury provides three types of best practices: Process Best Practices, Product Best Practices, and People Best Practices. Licensed customers of Mercury software can read and use the Mercury Best Practices available from the Customer Support site, http://support.mercury.com.

Documentation Updates

Mercury is continually updating its product documentation with new information. You can download the latest version of this document from the Customer Support Web site (http://support.mercury.com).

To download updated documentation:

1. In the Customer Support Web site, click the **Documentation** link.
2. Under **Please Select Product**, select **QuickTest Professional**.

   Note that if the required product does not appear in the list, you must add it to your customer profile. Click **My Account** to update your profile.
3. Click **Retrieve**. The Documentation page opens and lists the documentation available for the current release and for previous releases. If a document was updated recently, **Updated** appears next to the document name.
4. Click a document link to download the documentation.

Typographical Conventions

This guide uses the following typographical conventions:

UI Elements — 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."

Arguments — This style indicates method, property, or function arguments and book titles. For example, "Refer to the *Mercury 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."

Function_Name — This style indicates method or function names. For example, "The **wait_window** statement has the following parameters:"

[ ] — 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 5, “Working with Scripted Components.”

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 5, "Working with 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.

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 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 23, "Working with User-Defined Functions and Function Libraries."

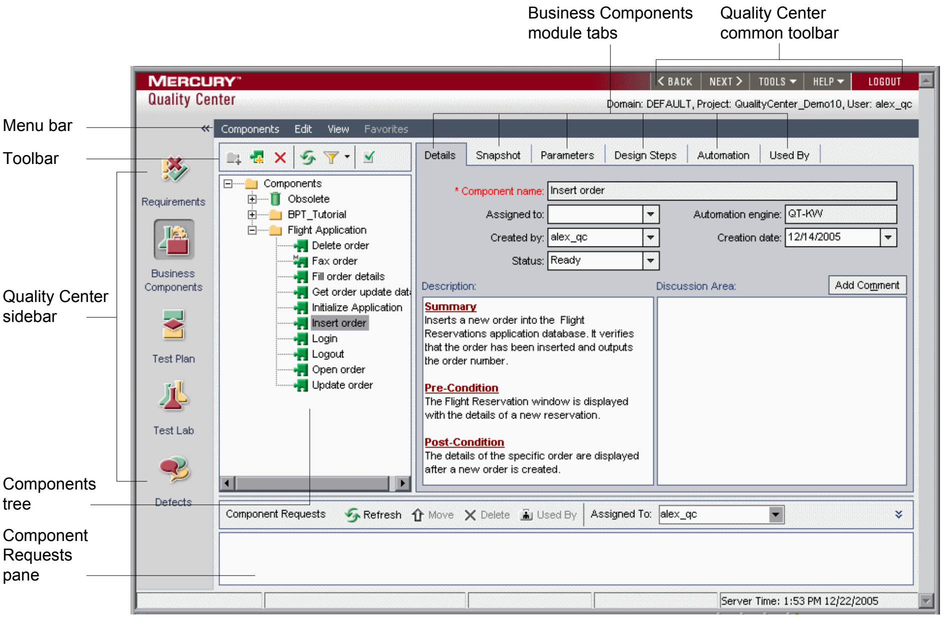
You also create, populate, and maintain shared object repository files that are used by QuickTest to identify the objects in your application. You define and modify test object information in shared object repositories using the QuickTest Object Repository Manager. After you associate shared object repository 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 14, "Working with Test Objects," and Chapter 16, "Managing Object Repositories."

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 3, "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 4, "Working with Business Components."

Creating Components in the Quality Center Business Components Module

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



Domain: DEFAULT, Project: QualityCenter_Demo10, User: alex_qc

* Component name: Insert order
Assigned to:
Automation engine: QT-KW
Created by: alex_qc
Creation date: 12/14/2005
Status: Ready

Summary
Inserts a new order into the Flight Reservations application database. It verifies that the order has been inserted and outputs the order number.

Pre-Condition
The Flight Reservation window is displayed with the details of a new reservation.

Post-Condition
The details of the specific order are displayed after a new order is created.

Assigned To: alex_qc

Server Time: 1:53 PM 12/22/2005

The Business Component module includes the following:

- **Details.** Provides a general summary of the component's purpose or goals, and the condition of the application before and after a component is run (its pre-conditions and post-conditions).
- **Snapshot.** Displays an image that provides a visual cue or description of the component's purpose or operations.
- **Parameters.** 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.** Enables you to create or view the manual steps of your business component, and to automate it if required.

- **Automation.** 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.** Provides details about the business process tests that include the currently selected business component. The tab also includes a link to the relevant business process test in the Test Plan module.
- **Component Requests pane.** Enables you to handle new component requests that were generated in the Test Plan module. Component requests are requests to add a new business component to the project.

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.

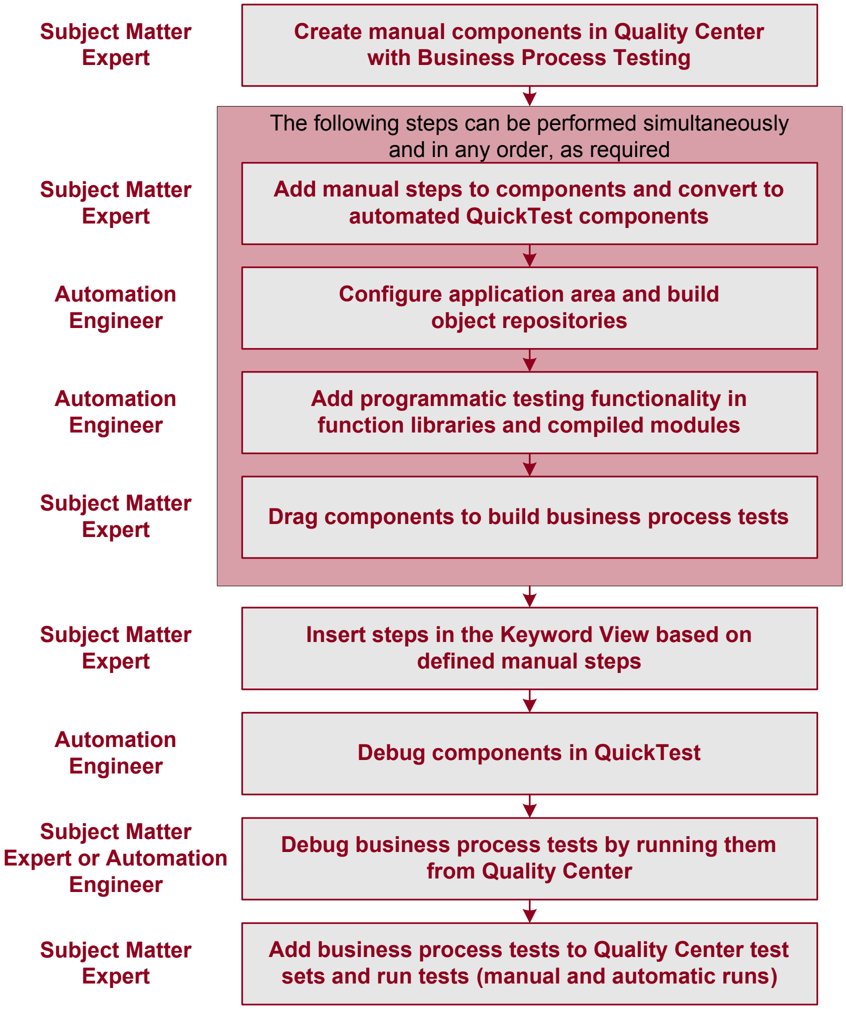
Note: When you run a business process test from Quality Center, the test run may also be influenced by settings in the QuickTest Remote Agent. For more information on the QuickTest Remote Agent, refer to the *QuickTest Professional User's Guide*.

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:


Subject Matter Expert
Create manual components in Quality Center with Business Process Testing
The following steps can be performed simultaneously and in any order, as required
Subject Matter Expert
Add manual steps to components and convert to automated QuickTest components
Automation Engineer
Configure application area and build object repositories
Automation Engineer
Add programmatic testing functionality in function libraries and compiled modules
Subject Matter Expert
Drag components to build business process tests
Subject Matter Expert
Insert steps in the Keyword View based on defined manual steps
Automation Engineer
Debug components in QuickTest
Subject Matter Expert or Automation Engineer
Debug business process tests by running them from Quality Center
Subject Matter Expert
Add business process tests to Quality Center test sets and run tests (manual and automatic runs)

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 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.mercury.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 14-day **demo** license
- 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 demo license, you can install a seat license, or you can choose to connect to a concurrent license server, if one is available on your network.

If needed, you can request a new seat license on the Mercury Customer Support Web site. The URL for the License Request Web site is http://support.mercury.com/license.

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

Updating the 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

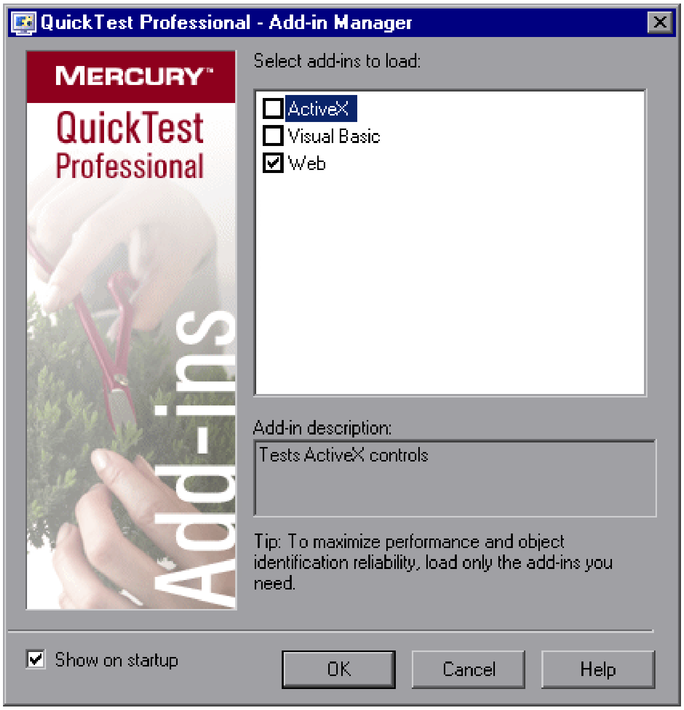
QuickTest at a Glance

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

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.



QuickTest Professional - Add-in Manager

Select add-ins to load:

□ ActiveX
□ Visual Basic
☑ Web

Add-in description:
Tests ActiveX controls

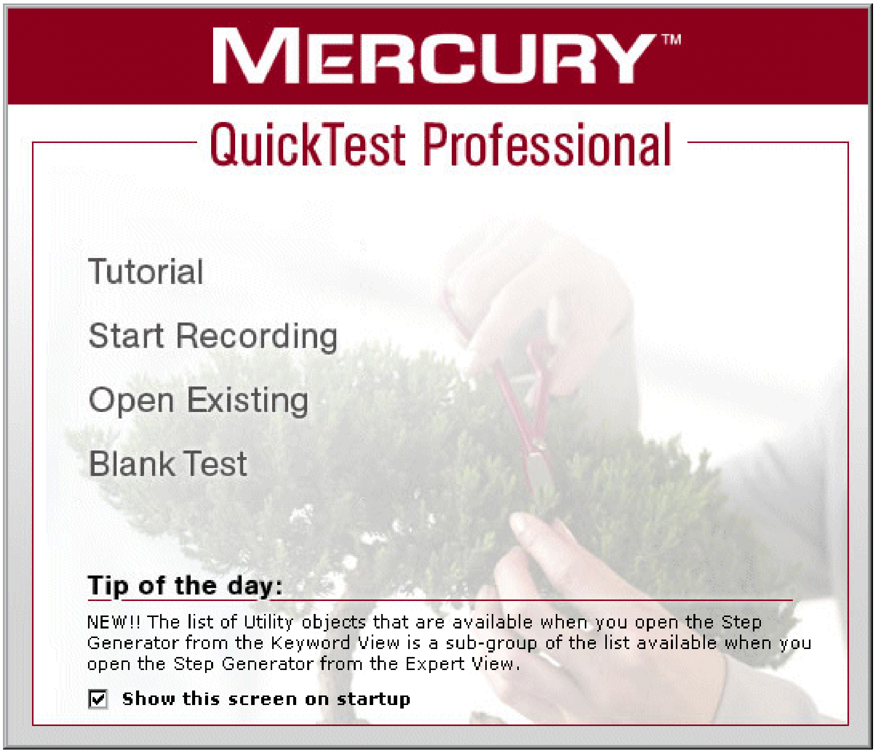
Tip: To maximize performance and object identification reliability, load only the add-ins you need.

☑ Show on startup

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

For more information on loading add-ins, see “Loading QuickTest Add-ins” on page 199.

Click **OK**. The QuickTest Professional window opens. You can choose to open the QuickTest tutorial, start recording a new test, open an existing test, or open a blank new test.



NEW!! The list of Utility objects that are available when you open the Step Generator from the Keyword View is a sub-group of the list available when you open the Step Generator from the Expert View.

☑ Show this screen on startup

Tips:

You can press the ESC key to close the window and open a blank test.

You can click **Tip of the Day** to browse through all the available tips.

If you do not want this window to be displayed the next time you start QuickTest, clear the **Show this screen on startup** check box.

Connecting to Your Quality Center Project

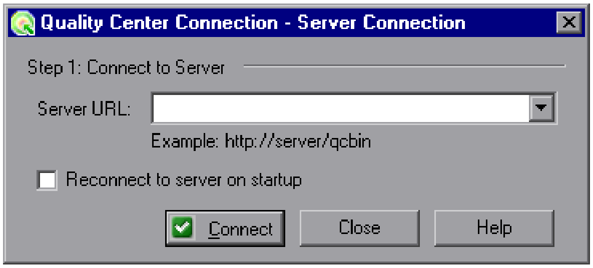
To work with business process testing, you must connect QuickTest to the Quality Center server on which your Quality Center project is stored. This server handles the connections between QuickTest and your Quality Center project. For Business Process Testing, you can connect from QuickTest Professional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

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.



Quality Center Connection - Server Connection
Step 1: Connect to Server
Server URL:
Example: http://server/qcbin
□ Reconnect to server on startup
Connect
Close
Help

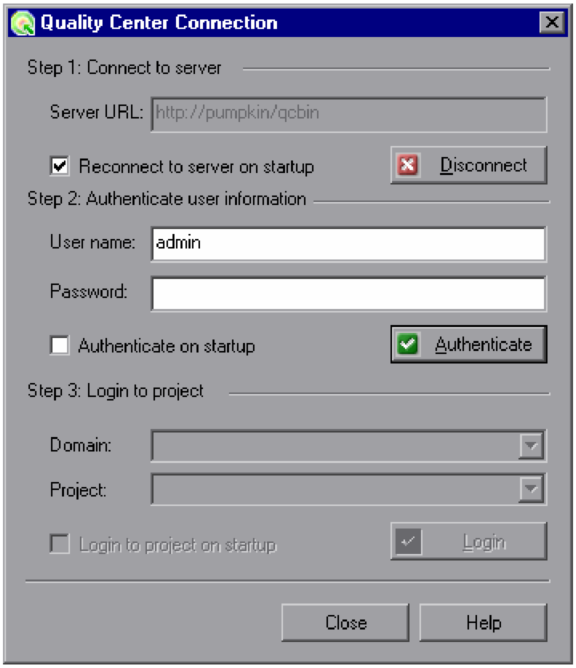
2 In the **Server URL** box, type the URL address of the Web server where Quality Center is installed.

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.



Quality Center Connection

Step 1: Connect to server

Server URL: http://pumpkin/qcbin

☑ Reconnect to server on startup

Step 2: Authenticate user information

User name: admin

Password:

□ Authenticate on startup

Step 3: Login to project

Domain:

Project:

□ Login to project on startup

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**.

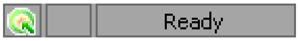
11 To automatically reconnect to the Quality Center server the next time you open QuickTest, select the **Reconnect to server on startup** check box.

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.



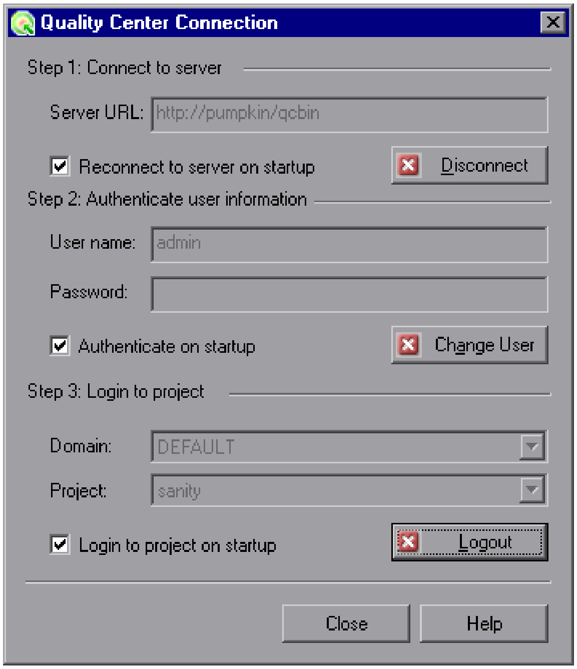
Tip: To view the current Quality Center connection, point to the **Quality Center** icon on the status bar. A tooltip displays the Quality Center server name and project to which QuickTest is connected. To open the Quality Center Connection dialog box, double-click the **Quality Center** icon.

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.



Step 1: Connect to server

Server URL: http://pumpkin/qcbin

☑ Reconnect to server on startup

Step 2: Authenticate user information

User name: admin

Password:

☑ Authenticate on startup

Step 3: Login to project

Domain: DEFAULT

Project: sanity

☑ Login to project on startup

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 document(s) 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 46.

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

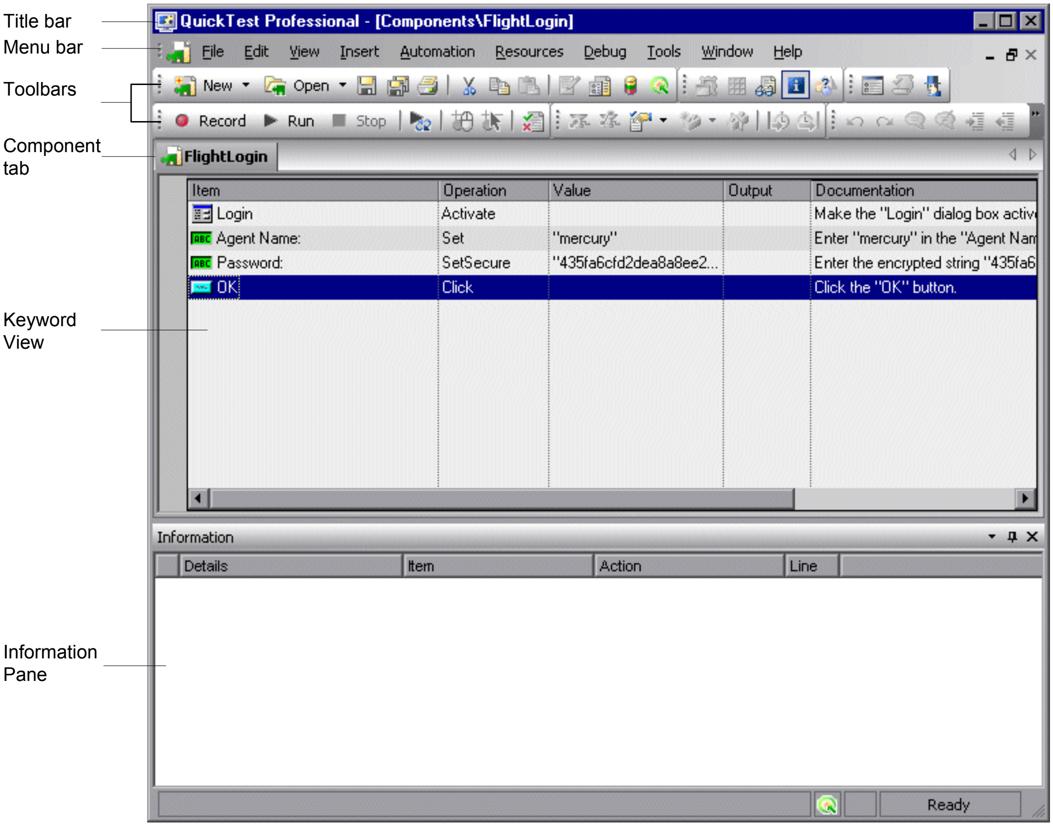
- **Business Component.** Enables you to create, view, and modify your business component using keywords and operations. For more information, see Chapter 6, "Working with the Keyword View."
- **Scripted Component.** Enables you to create, view, and modify your scripted component in Keyword View or Expert View (described below). For more information on scripted components, see Chapter 5, "Working with Scripted Components." For more information on the Expert View, refer to the *Mercury QuickTest Professional User's Guide*.
- **Application Area.** Enables you to define resources and settings for your components. For more information, see Chapter 3, "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 23, "Working with User-Defined Functions and Function Libraries."

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.
- **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 6, "Working with the Keyword View."
- **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. (Not displayed by default)
- **Debug Viewer pane.** Assists you in debugging your document. The Debug Viewer pane contains the **Watch**, **Variables**, and **Command** tabs. (Not displayed by default)
- **Status bar.** Displays the status of the QuickTest application and other relevant information.

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 “Customizing the QuickTest Window Layout” on page 37.



Item	Operation	Value	Output	Documentation
Login	Activate			Make the "Login" dialog box activ
Agent Name:	Set	"mercury"		Enter "mercury" in the "Agent Nam
Password:	SetSecure	"435fa6cfd2dea8a8ee2...		Enter the encrypted string "435fa6
OK	Click			Click the "OK" button.

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.

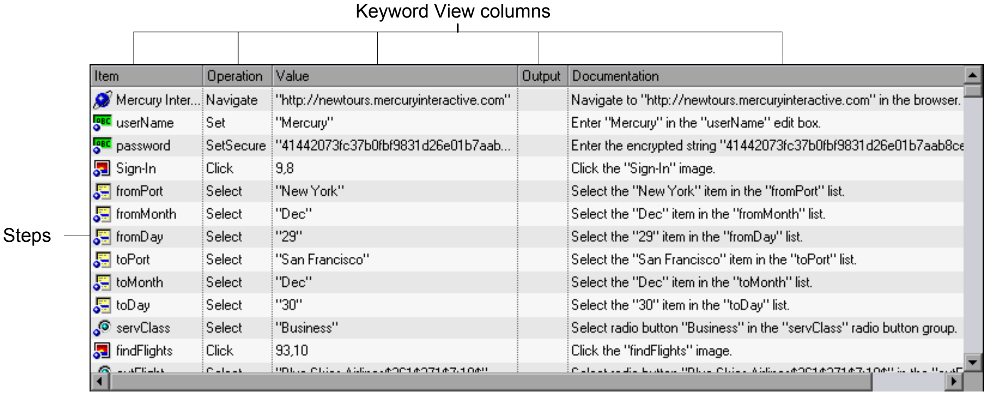
Note: You can apply the Microsoft Windows XP theme to the QuickTest window only if your computer is set to use a Windows XP theme.

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 546.

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.



Item	Operation	Value	Output	Documentation
Mercury Inter...	Navigate	"http://newtours.mercuryinteractive.com"		Navigate to "http://newtours.mercuryinteractive.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.

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:

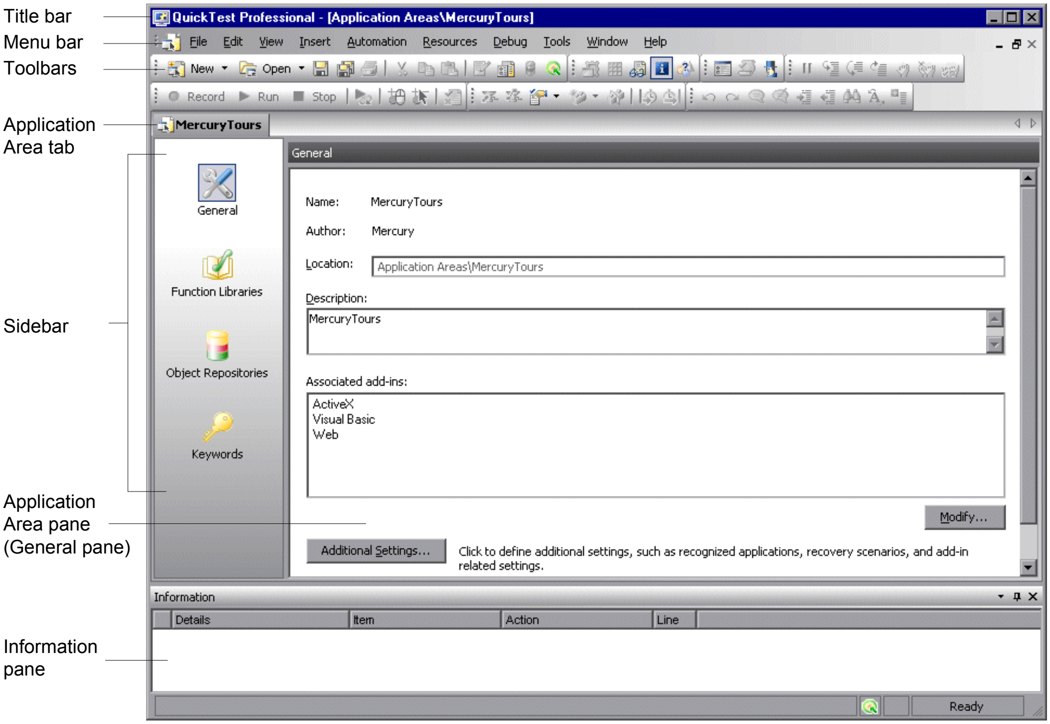
General. Displays general information about the application area and enables you to modify its general settings, such as specifying associated add-ins, recovery scenarios, and other settings.

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 3, "Working with Application Areas."



Name: MercuryTours

Author: Mercury

Location: Application Areas\MercuryTours

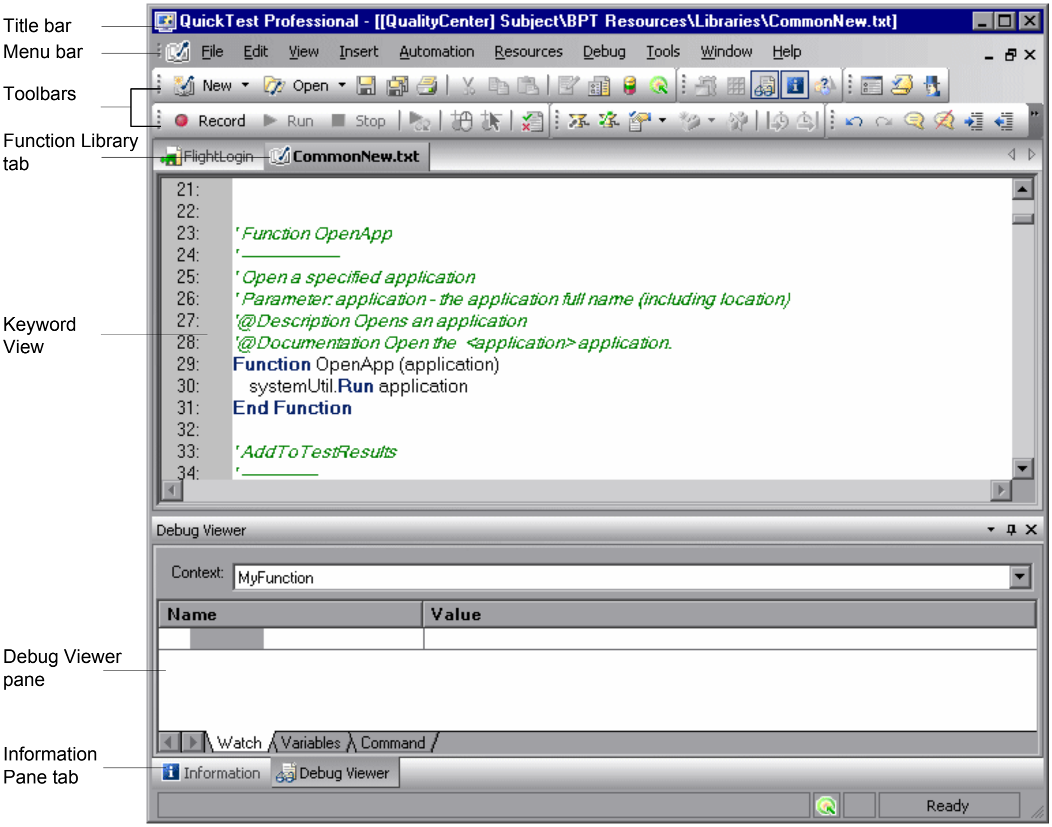
Description:
MercuryTours

Associated add-ins:
ActiveX
Visual Basic
Web

Click to define additional settings, such as recognized applications, recovery scenarios, and add-in related settings.

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 23, “Working with User-Defined Functions and Function Libraries.”



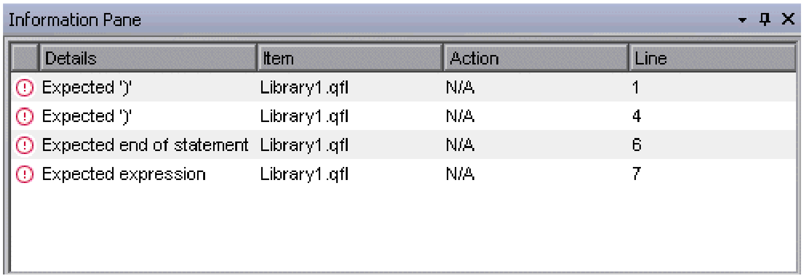
```


'Function OpenApp
'____________
'Open a specified application
'Parameter: application - the application full name (including location)
'@Description Opens an application
'@Documentation Open the <application> application.
Function OpenApp (application)
  systemUtil.Run application
End Function

'AddToTestResults
'__________
```

Context: MyFunction

Name	Value

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.

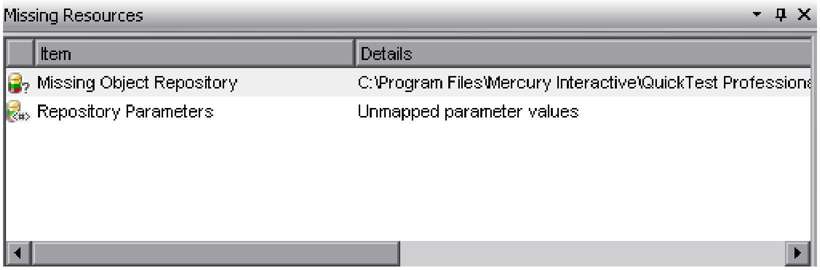


Details	Item	Action	Line
Expected ')'	Library1.qfl	N/A	1
Expected ')'	Library1.qfl	N/A	4
Expected end of statement	Library1.qfl	N/A	6
Expected expression	Library1.qfl	N/A	7

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 678.

Missing Resources Pane

The Missing Resources pane provides a list of the resources that are specified in your test but cannot be found, such as 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.



Item	Details
Missing Object Repository	C:\Program Files\Mercury Interactive\QuickTest Professiona
Repository Parameters	Unmapped parameter values

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 191.

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, click the **Debug Viewer** button or 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 19, "Debugging Components and Function Libraries."

Customizing the QuickTest Window Layout

You can customize the layout of the QuickTest window, and you can restore the default layout. When customizing the window, you can move and resize panes, select to show or auto-hide panes, create tabbed panes, select which toolbars to display, and so forth.

Note: When customizing or restoring the QuickTest window layout, the layout is customized or restored for all document 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.

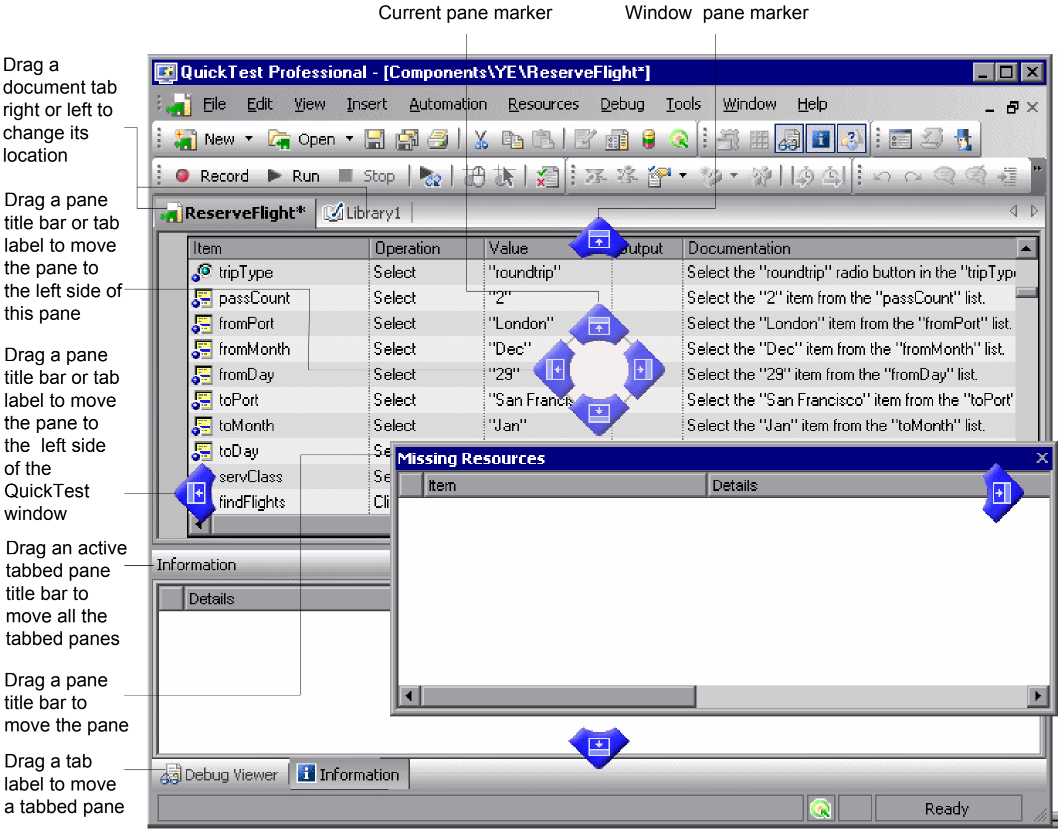
Tip: To move a dockable pane without snapping it into place, press CTRL while dragging it to the required location.

To move panes:

1 In the QuickTest window, drag the title bar or tab of the pane you want to move.

Tip: 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.



Current pane marker

Window pane marker

Drag a document tab right or left to change its location

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

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

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

Drag a pane title bar to move the pane

Drag a tab label to move a tabbed pane

QuickTest Professional - [Components\YE\ReserveFlight*]

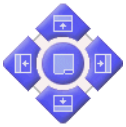
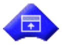
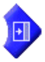
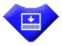
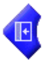
Item	Operation	Value	Output	Documentation
tripType	Select	"roundtrip"		Select the "roundtrip" radio button in the "tripType
passCount	Select	"2"		Select the "2" item from the "passCount" list.
fromPort	Select	"London"		Select the "London" item from the "fromPort" list.
fromMonth	Select	"Dec"		Select the "Dec" item from the "fromMonth" list.
fromDay	Select	"29"		Select the "29" item from the "fromDay" list.
toPort	Select	"San Franci		Select the "San Francisco" item from the "toPort"
toMonth	Select	"Jan"		Select the "Jan" item from the "toMonth" list.
toDay	Se			
servClass	Se			
findFlights	Cli			

Tips:

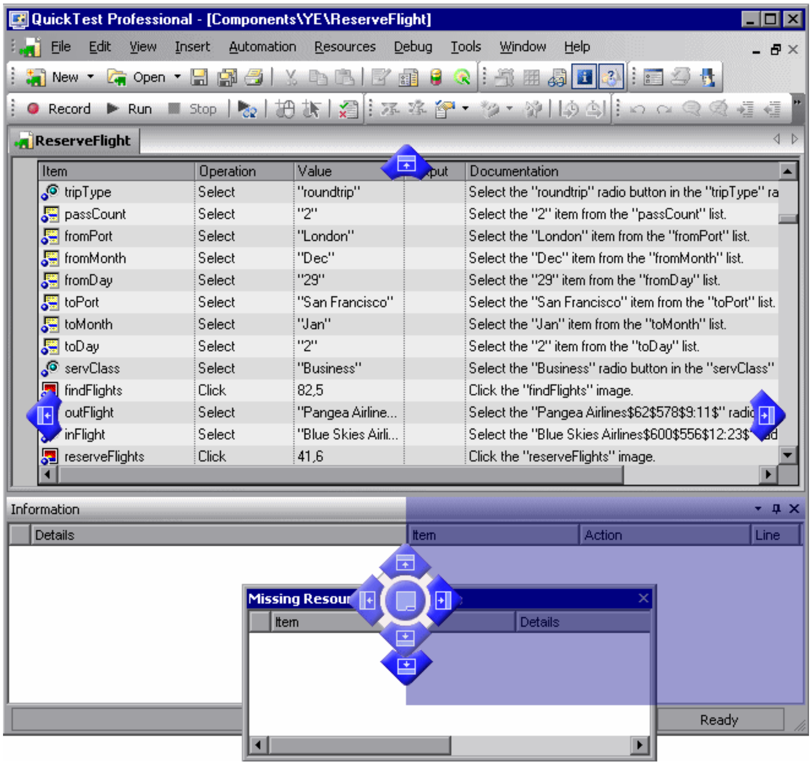
To move a single tabbed pane, drag the tab label. Once you start dragging the tabbed pane, the tab is removed, and its title bar is displayed.

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

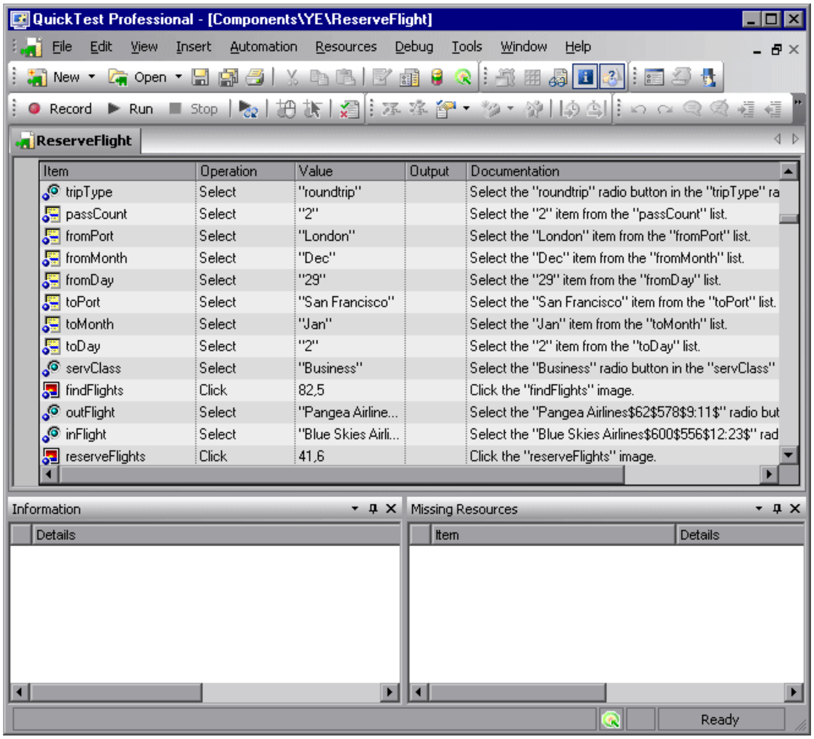


Item	Operation	Value	put	Documentation
tripType	Select	"roundtrip"		Select the "roundtrip" radio button in the "tripType" ra
passCount	Select	"2"		Select the "2" item from the "passCount" list.
fromPort	Select	"London"		Select the "London" item from the "fromPort" list.
fromMonth	Select	"Dec"		Select the "Dec" item from the "fromMonth" list.
fromDay	Select	"29"		Select the "29" item from the "fromDay" list.
toPort	Select	"San Francisco"		Select the "San Francisco" item from the "toPort" list.
toMonth	Select	"Jan"		Select the "Jan" item from the "toMonth" list.
toDay	Select	"2"		Select the "2" item from the "toDay" list.
servClass	Select	"Business"		Select the "Business" radio button in the "servClass"
findFlights	Click	82,5		Click the "findFlights" image.
outFlight	Select	"Pangea Airline...		Select the "Pangea Airlines$62$578$9:11$" radic
inFlight	Select	"Blue Skies Airli...		Select the "Blue Skies Airlines$600$556$12:23$" ad
reserveFlights	Click	41,6		Click the "reserveFlights" image.

Details	Item	Action	Line

Item	Details

3 Release the mouse button. The Missing Resources pane snaps into place and is displayed as a new pane in the shaded area.



Item	Operation	Value	Output	Documentation
tripType	Select	"roundtrip"		Select the "roundtrip" radio button in the "tripType" ra
passCount	Select	"2"		Select the "2" item from the "passCount" list.
fromPort	Select	"London"		Select the "London" item from the "fromPort" list.
fromMonth	Select	"Dec"		Select the "Dec" item from the "fromMonth" list.
fromDay	Select	"29"		Select the "29" item from the "fromDay" list.
toPort	Select	"San Francisco"		Select the "San Francisco" item from the "toPort" list.
toMonth	Select	"Jan"		Select the "Jan" item from the "toMonth" list.
toDay	Select	"2"		Select the "2" item from the "toDay" list.
servClass	Select	"Business"		Select the "Business" radio button in the "servClass"
findFlights	Click	82,5		Click the "findFlights" image.
outFlight	Select	"Pangea Airline...		Select the "Pangea Airlines$62$578$9:11$" radio but
inFlight	Select	"Blue Skies Airli...		Select the "Blue Skies Airlines$600$556$12:23$" rad
reserveFlights	Click	41,6		Click the "reserveFlights" image.

Details

Item	Details

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 42.

---

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: • **Floating.** Opens the pane on top of all the other windows and panes, with its own title bar • **Docking.** Docks the pane to the QuickTest window. • **Auto-hide.** Displays the pane as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window. • **Hide.** Closes the pane.
📌	The **Auto Hide** button hides the pane. The pane is displayed as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window. To display the pane, hold the cursor over the side-tab. The button toggles to the **Dock** button, shown below.

Button	Description
📌	The **Dock** button docks the pane to the QuickTest window. The pane returns the position it was in before it was hidden, and the button toggles to the **Auto Hide** button, shown above.
☒	The **Close** button closes the pane. The pane is removed from the QuickTest window. To reopen the pane, select it from the **View** menu. **Tip:** You can also close a pane by right-clicking and choosing **Hide** from the context menu.

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.

Note: For more information on the Options dialog box, see Chapter 9, “Setting Global Testing Options.”

Working With Multiple Documents

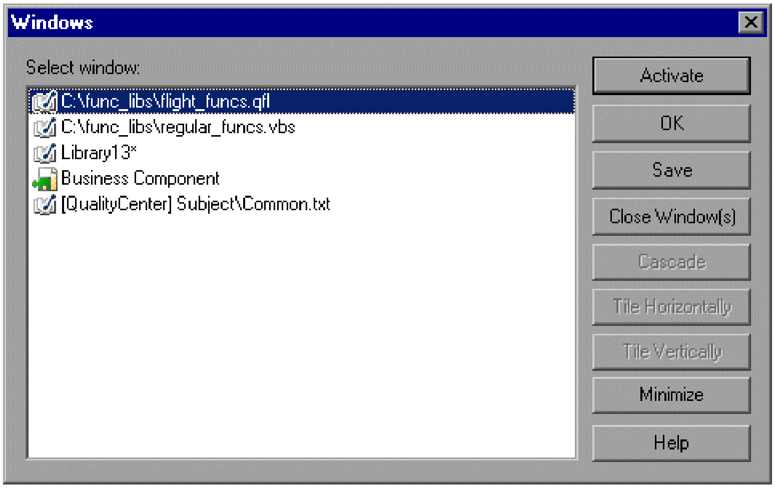
QuickTest enables you to open and work on one component or application area at a time. In addition, you can open and work on multiple function libraries simultaneously. You can open any function library, regardless of whether it is associated with the currently open component or application area.

The **Window** 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.



Select window:

C:\func_libs\flight_funcs.qfl
C:\func_libs\regular_funcs.vbs
Library13*
Business Component
[QualityCenter] Subject\Common.txt

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 document(s).
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.

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.

Choosing Commands from a Menu

You can select most QuickTest commands from the menus in the menu bar. Shortcut keys, where available, are displayed with the corresponding menu commands. For more information, see "Performing Commands Using Shortcut Keys" on page 52.

Clicking Commands on a Toolbar

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

Notes:

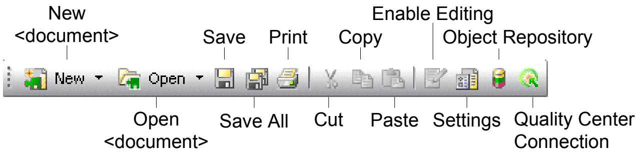
Not all toolbars are relevant for all document types. Only those toolbars that are relevant for components, application areas, and function libraries are described here.

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 more information on working with business components, see Chapter 4, "Working with Business Components." For more information on working with scripted components, see Chapter 5, "Working with Scripted Components." For more information on working with application areas, see Chapter 3, "Working with Application Areas." For more information on working with function libraries, see Chapter 23, "Working with User-Defined Functions and Function Libraries."

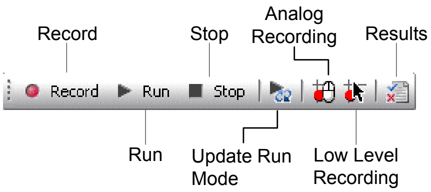
The following buttons are displayed on the **Standard** toolbar:



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.

Automation Toolbar

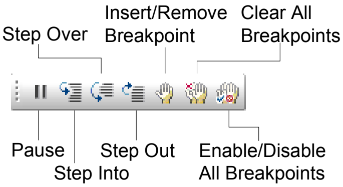
The **Automation** toolbar contains buttons for recording and running your component. The following buttons are displayed on the **Automation** toolbar:



Note: The **Analog Recording** and **Low Level Recording** buttons apply only to tests.

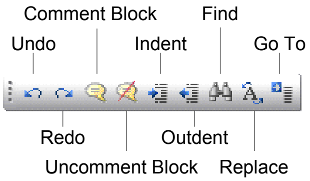
Debug Toolbar

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



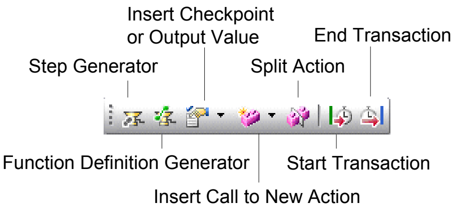
Edit Toolbar

The **Edit** toolbar contains buttons for the commands used when editing your function library. The following buttons are displayed on the **Edit** toolbar:



Insert Toolbar

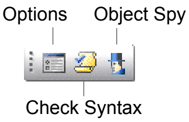
The **Insert** toolbar contains buttons for the commands used when working with function libraries. The following buttons are displayed on the **Insert** toolbar:



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

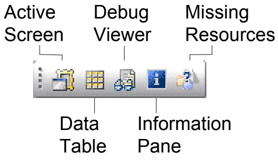
The **Tools** toolbar contains buttons for the commands used to access tools that assist you when working with your test. The following buttons are displayed on the **Tools** toolbar:



Note: 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. The following buttons are displayed on the **View** toolbar.



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

Performing Commands Using Shortcut Keys

You can perform some QuickTest commands by pressing shortcut keys. The shortcut keys listed below are displayed on the corresponding menu commands.

You can perform the following **File** menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
New > Test	CTRL+N	Creates a new test.
New > Business Component	CTRL+SHIFT+N	Creates a new business 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.

Command	**Shortcut Key**	**Function**
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.
Save	CTRL+S	Saves the active document.
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.

You can perform the following **Edit** menu commands by pressing the corresponding shortcut keys:

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.

Command	Shortcut Key	Function
Delete	DELETE	Deletes the selection from your document.
Step Properties > Comment Properties	CTRL+ENTER; ALT+ENTER	Displays the Comment Properties dialog box of a selected object.
Step Properties > Object Properties	CTRL+ENTER; ALT+ENTER	Displays the Object Properties dialog box of a selected object.
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.
Advanced > Uncomment Block	CTRL+SHIFT+M	Removes the comment formatting from the current or selected rows.
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.

You can perform the following **Insert** menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
Checkpoint > Standard Checkpoint	F12	Creates a standard checkpoint for an object or a table.
Output Value > Standard Output Value	CTRL+F12	Creates a standard output value for an object or a table.
Step Generator	F7	Opens the Step Generator.
New Step	F8; INSERT	Inserts a new step in the Keyword View.
New Step After Block	SHIFT+F8	Inserts a new step after a conditional or loop block in the Keyword View.

You can perform the following **Automation** menu commands by pressing the corresponding shortcut keys:

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 from Step	CTRL+F5	Starts a run session from the selected step.
Analog Recording	SHIFT+ALT+F3	Starts recording in analog recording mode.
Low Level Recording	CTRL+SHIFT+F3	Starts recording in low level recording mode.

You can perform the following **Resources** menu command by pressing the corresponding shortcut key:

Command	Shortcut Key	Function
Object Repository	CTRL+R	Opens the Object Repository dialog box.

You can perform the following **Debug** menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
Step Into	F11	Runs only the current line of the script. If the current line calls a method, the method is displayed in the view but is not performed.
Step Over	F10	Runs only the current line of the script. When the current line calls a method, the method is performed in its entirety, but is not displayed in the view.
Step Out	SHIFT+F11	Runs to the end of the method then pauses the run session. (Available only after running a method using **Step Into**.)
Run to Step	CTRL+F10	Runs until the current step.
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.

You can perform the following **Tools** menu command by pressing the corresponding shortcut key:

Command	**Shortcut Key**	**Function**
Check Syntax	CTRL+7	Checks the syntax of the active document.

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.

Command	Shortcut Key	Function
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.
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.

You can perform the following special options using shortcut keys:

Option	Shortcut Key	Function
Switch between Keyword View and Expert View	CTRL+PAGE UP/PAGE DOWN	Toggles between the Keyword View and Expert View.
Switch between open documents	CTRL+TAB	Changes the display to another open document type.
Open context menu	SHIFT+F10, or press the Application Key (▤) *[Microsoft Natural Keyboard only]*	Opens the context menu for the selected step data cell in the Data Table.

Option	**Shortcut Key**	**Function**
Expand all branches	* *[on the numeric keypad]*	Expands all branches in the Keyword View.
Expand branch	+ *[on the numeric keypad]*	Expands the selected item branch and all branches below it in the Keyword View.
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 performed an upgrade installation, or 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 external add-ins installed, you may have items in your program folder that relate specifically to these add-ins.

- **Add-ins.** Provides links to Readme documentation for any external QuickTest add-ins installed on your computer.
- **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 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 virtually every QuickTest feature and capability. 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.
  - **QuickTest Professional Help.** Opens a comprehensive help file containing the *QuickTest Professional User's Guide*, the *QuickTest Professional for Business Process Testing User's Guide*, the corresponding user's guide for each installed add-in (if any), the *QuickTest Professional Object Model Reference* (including the relevant sections for any installed add-ins), 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 Professional Code Samples Plus.** Opens the QuickTest Professional Code Samples Plus Help, which in previous versions provided answers to frequently asked questions, tips and tricks, sample function libraries, and code and SDK samples. The utilities, features, and information that were available using QuickTest Professional Code Samples Plus have been incorporated into the main QuickTest Professional application.

- **Sample Applications.** Contains the following links to sample applications that you can use to practice testing with QuickTest:
  - **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, refer to the *Mercury QuickTest Professional Tutorial*.
  - **Flight.** Opens a sample flight reservation Windows application. To access the application, enter any username and the password **mercury**.
- **Tools.** Contains the following utilities and tools that assist you with the testing process:
  - **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 166.
  - **Remote Agent.** Activates the QuickTest Remote Agent, which determines how QuickTest behaves when a test is run by a remote application such as Quality Center. For more information, refer to the *Mercury 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, refer to the *Mercury 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.
  - **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 563.
  - **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, click the **Help** button in the Action Conversion Tool window.

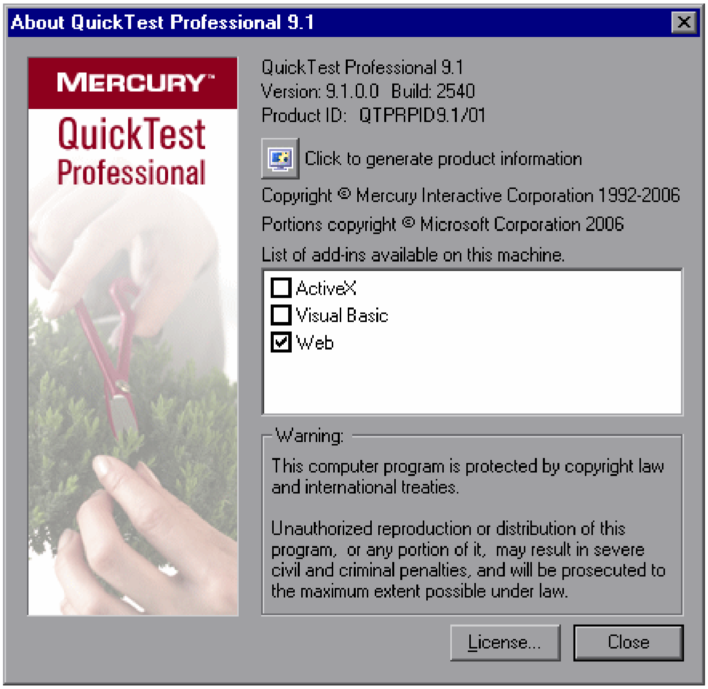
  - **License Validation Utility.** Opens the License Validation utility, which enables you to retrieve and validate license information. For more information, see "Validating QuickTest Licenses" on page 917.
  - **Register New Browser Control.** Opens the Register Browser Control Utility, which enables you to register your browser control application so that QuickTest Professional recognizes your Web object when recording or running tests. For more information, see "Registering Browser Controls" on page 852.
  - **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 in a batch all the QuickTest business components in a version 9.0 or later Quality Center project from an earlier version to the current format. For more information, click the **Help** button in the Business Component Upgrade Tool window.
  - **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, refer to the *Mercury QuickTest Professional User's Guide*.
- **Check for Updates.** Checks online for any available updates to QuickTest Professional. You can choose which updates you want to download and (optionally) install. For more information, refer to the *Mercury QuickTest Professional User's Guide*.
- **QuickTest Professional**. Opens the QuickTest Professional application.
- **Readme**. Opens the QuickTest Professional Readme, which provides the latest news and information on QuickTest Professional.
- **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 "Understanding the Test Results Window" on page 543.
- **Uninstall QuickTest Professional.** Uninstalls QuickTest Professional and all of its components, including built-in and external add-ins. Refer to the *Mercury QuickTest Professional Installation Guide* for more information.

Viewing Product Information

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

To view the product information:

1 In QuickTest, choose **Help** > **About QuickTest Professional**. The About QuickTest Professional 9.1 window opens.



About QuickTest Professional 9.1

QuickTest Professional 9.1
Version: 9.1.0.0 Build: 2540
Product ID: QTPRPID9.1/01

Click to generate product information

Copyright © Mercury Interactive Corporation 1992-2006

Portions copyright © Microsoft Corporation 2006

List of add-ins available on this machine.

□ ActiveX
□ Visual Basic
☑ Web

Warning:

This computer program is protected by copyright law and international treaties.

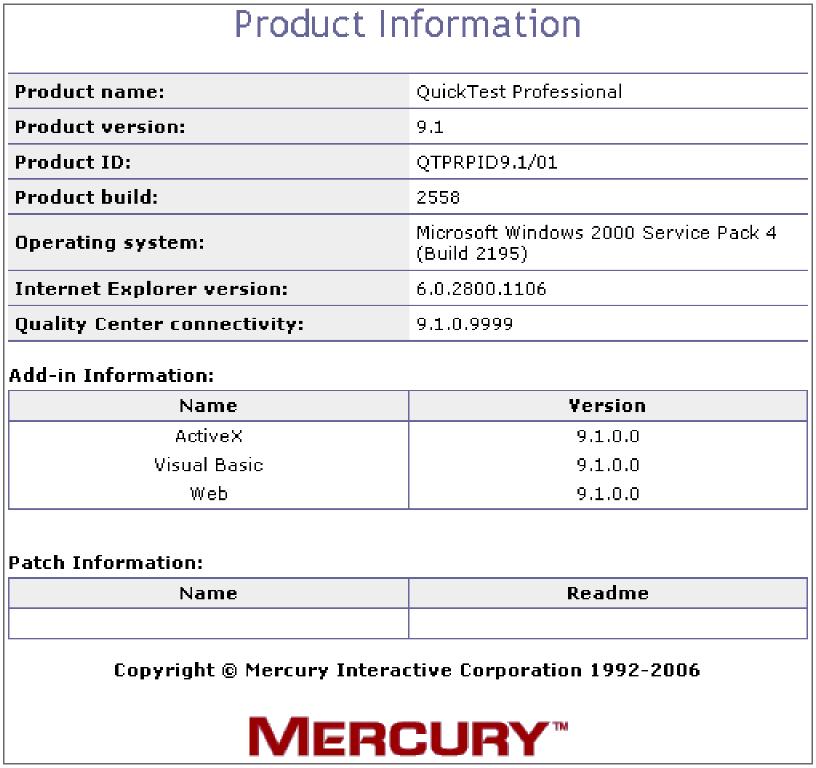
Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

The About QuickTest Professional 9.1 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 “Selecting QuickTest Add-Ins” on page 197.

Tip: To view details for, or modify, the QuickTest Professional licenses installed on your computer, click the **License** button. For more information, refer to the *Mercury 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:	9.1
Product ID:	QTPRPID9.1/01
Product build:	2558
Operating system:	Microsoft Windows 2000 Service Pack 4 (Build 2195)
Internet Explorer version:	6.0.2800.1106
Quality Center connectivity:	9.1.0.9999

Add-in Information:

Name	Version
ActiveX	9.1.0.0
Visual Basic	9.1.0.0
Web	9.1.0.0

Patch Information:

Name	Readme

Copyright © Mercury Interactive Corporation 1992-2006

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, and their version and build numbers.
- **Patch Information.** The names of any QuickTest patches installed on your computer, and links to their readme files.

Part II

Working with Application Areas and Components

3

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.

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 a 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 *Business Process Testing User's Guide*.

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**.

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”.

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 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 131.

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 22.

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 108.

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. You can also specify the Windows-based applications on which a component associated with the application area can record and run, set the browser time-out period, and 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 77.
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 83.
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 88.
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 95.

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 22.

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 9.1 on a computer running QuickTest 8.2.

To open an application area:

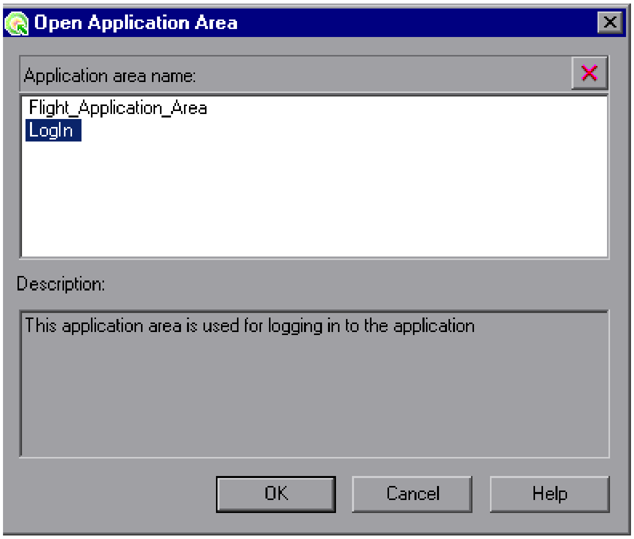
1 View the application areas connected to the current Quality Center project.

➤ 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.


Open Application Area

Application area name:

Flight_Application_Area
LogIn

Description:

This application area is used for logging in to the application

2 Select an application area and click **OK**. The selected application area opens.

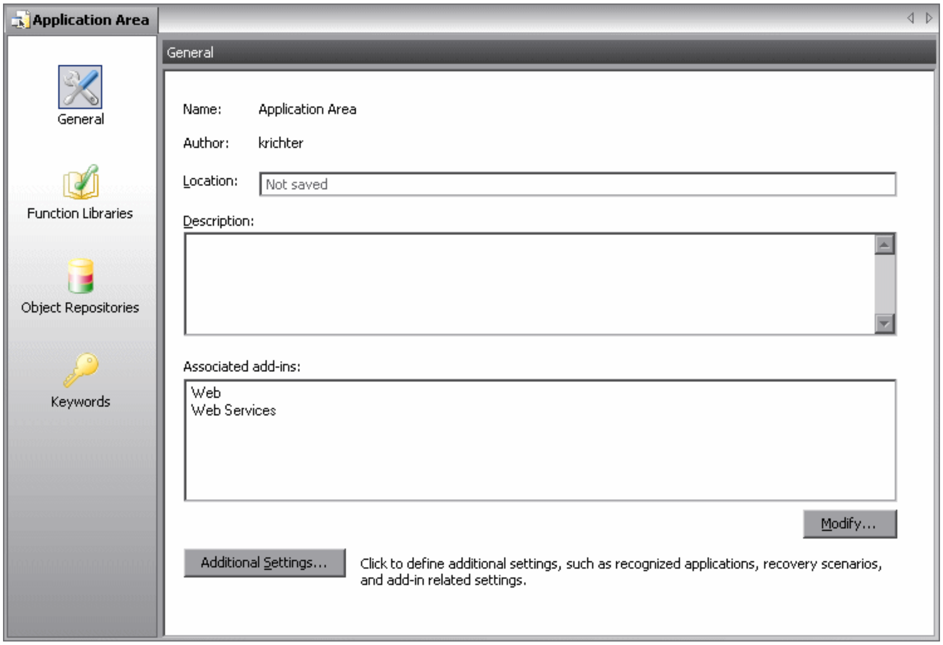
You can now view and modify the settings for the application area. For more information, see:

- "Defining General Settings" on page 77
- "Managing Function Libraries" on page 83
- "Managing Shared Object Repositories" on page 88
- "Managing Keywords" on page 95
- "Defining Additional Settings" on page 99

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 110.

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.



Name: Application Area

Author: krichter

Location: Not saved

Associated add-ins:
Web
Web Services

Click to define additional settings, such as recognized applications, recovery scenarios, and add-in related settings.

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 108.
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. 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 *Business Process Testing User's Guide*. 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**. **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 108.
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. **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. For more information, see "Associating Add-ins with Your Component" on page 80.

Item	Description
Additional Settings button	Opens the Application Area Settings dialog box (described on page 99), which is divided into several tabs. • **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 99. • **Web.** Enables you to set the time-out period for browser navigations. For more information, see "Defining Web Settings for Your Application Area" on page 103. • **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 104. The Application Area Settings dialog box may also contain additional tabs corresponding to any external add-ins that are loaded, for example, Java or SAP. For information on tabs related to external add-ins, refer to the relevant QuickTest add-in documentation.
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 80.

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.

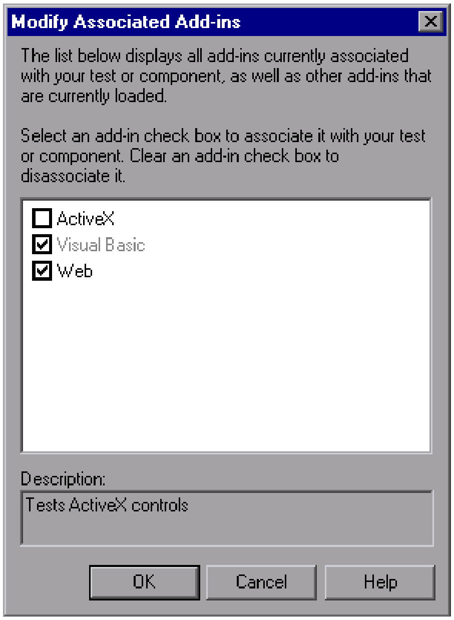
Choosing to associate an add-in with an application area instructs QuickTest to check that the associated add-in is loaded each time you open a component that is associated with that application area. When you create a new component, its associated add-ins are those defined in the component's application 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.



Modify Associated Add-ins

The list below displays all add-ins currently associated with your test or component, as well as other add-ins that are currently loaded.

Select an add-in check box to associate it with your test or component. Clear an add-in check box to disassociate it.

- ☐ ActiveX
- ☑ Visual Basic
- ☑ Web

Description:

Tests ActiveX controls

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.

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.

In the above example:

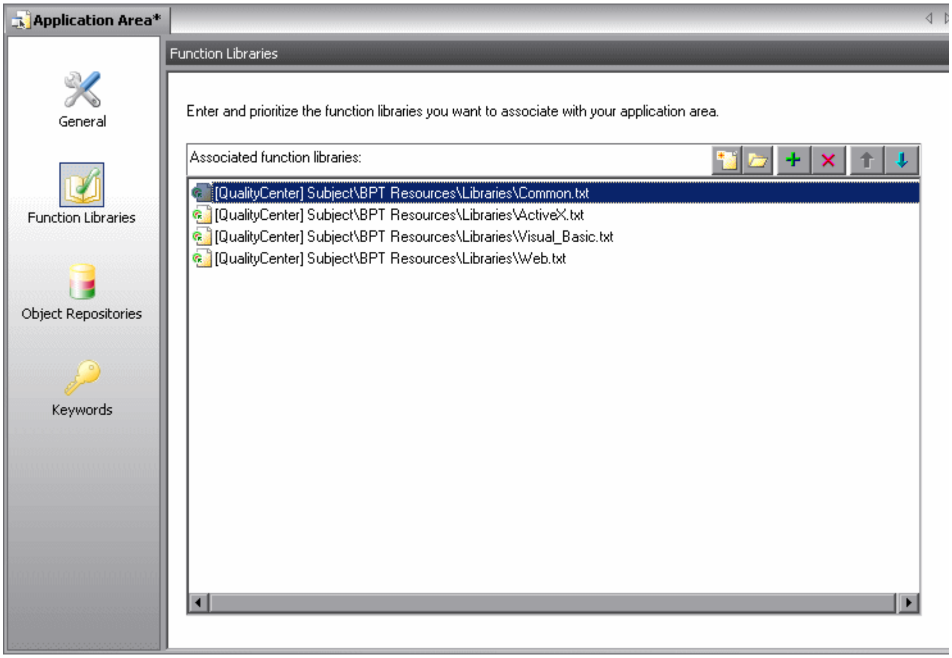
- Web is loaded and associated with the application area.
- ActiveX is loaded, but not associated with the application area.
- Visual Basic is associated with the application area, but is not loaded.

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 216.

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.



Enter and prioritize the function libraries you want to associate with your application area.

Associated function libraries:

[QualityCenter] Subject\BPT Resources\Libraries\Common.txt
[QualityCenter] Subject\BPT Resources\Libraries\ActiveX.txt
[QualityCenter] Subject\BPT Resources\Libraries\Visual_Basic.txt
[QualityCenter] Subject\BPT Resources\Libraries\Web.txt

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 23, "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 23, "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 85.
✕	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**

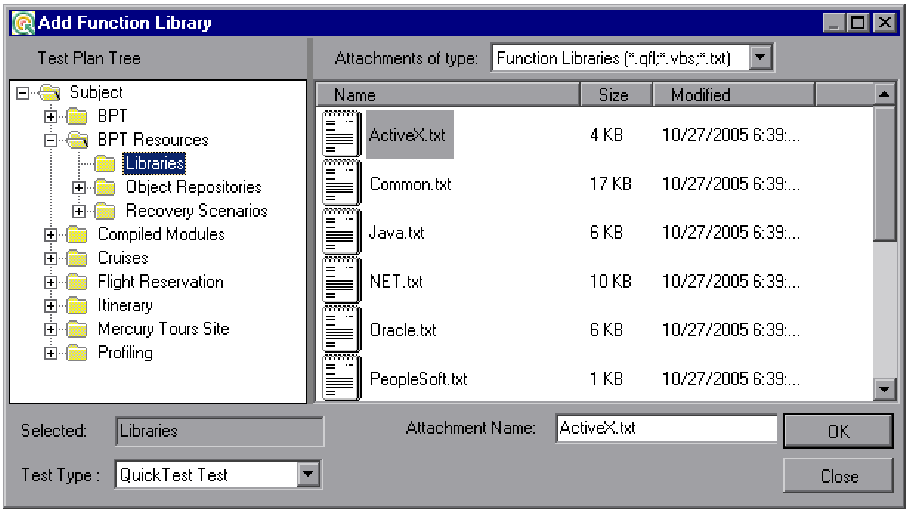
Tip: If another application area is already open, you can click the **Open** button and then select the required application area.

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.



Attachments of type: Function Libraries (*.qfl;*.vbs;*.txt)

Name	Size	Modified
ActiveX.txt	4 KB	10/27/2005 6:39:...
Common.txt	17 KB	10/27/2005 6:39:...
Java.txt	6 KB	10/27/2005 6:39:...
NET.txt	10 KB	10/27/2005 6:39:...
Oracle.txt	6 KB	10/27/2005 6:39:...
PeopleSoft.txt	1 KB	10/27/2005 6:39:...

Selected: Libraries

Attachment Name: ActiveX.txt

Test Type : QuickTest Test

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.

---

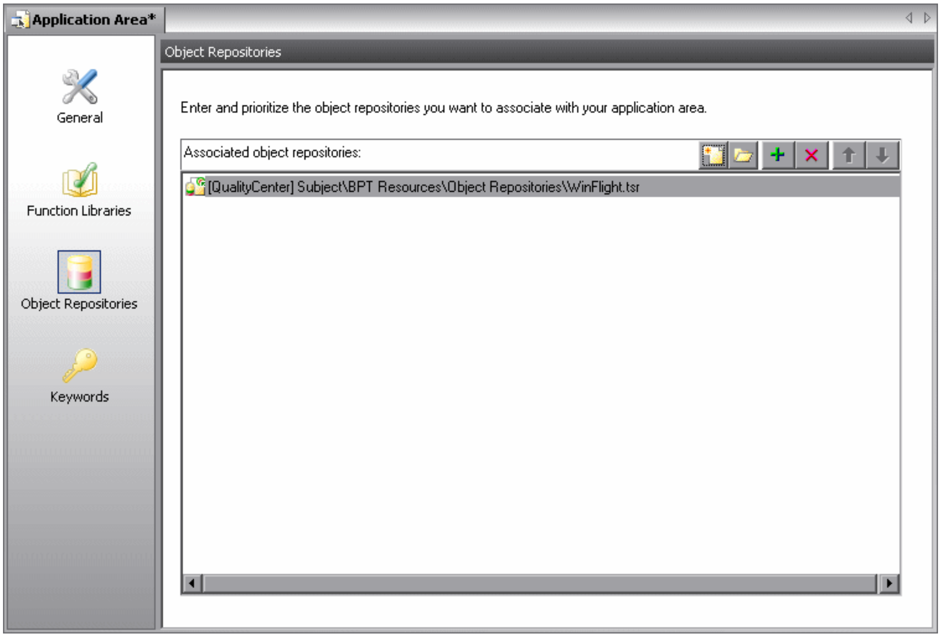
6 If you want to add content to the new function library or modify the file directly from QuickTest, select the file in the Function Libraries pane and click the **Open Function Library** button or double-click the function library in the list. The file opens in a function library window and can be edited 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 23, "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.



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

Associated object repositories:

[QualityCenter] Subject\BPT Resources\Object Repositories\WinFlight.tsr

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 16, "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 91.

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 154.

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 346.

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 16, “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 93.
✕	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 16, "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**

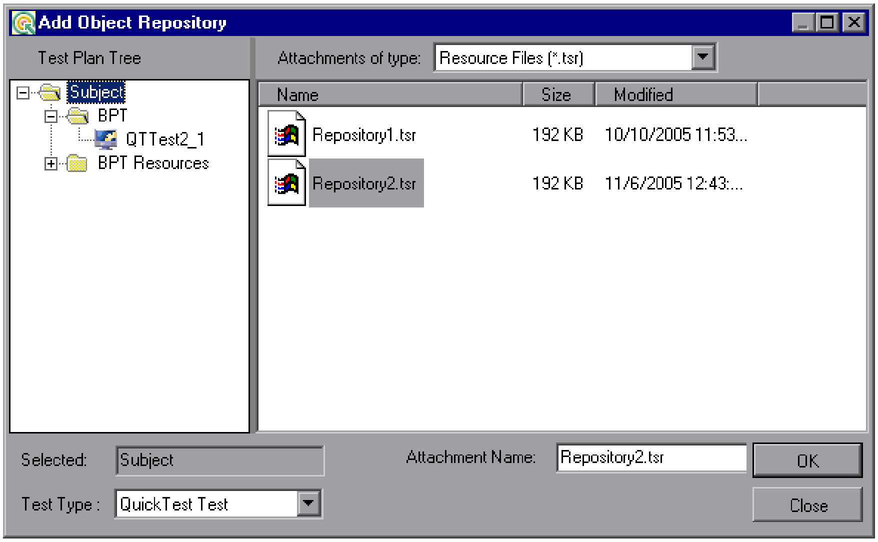
Tip: If another application area is already open, you can also click the **Open** button and then select the application area you require.

---

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.



Attachments of type: Resource Files (*.tsr)

Name	Size	Modified
Repository1.tsr	192 KB	10/10/2005 11:53...
Repository2.tsr	192 KB	11/6/2005 12:43:...

Selected: Subject

Test Type : QuickTest Test

Attachment Name: Repository2.tsr

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.

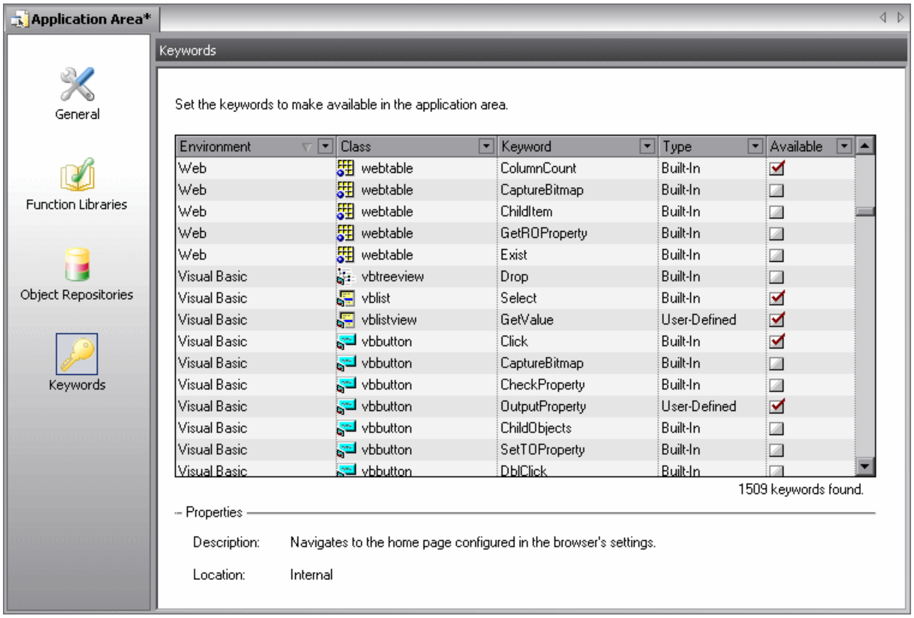
8 If you want to rename the object repository, you can click it twice, or select it and press F2.

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.



Keywords

Set the keywords to make available in the application area.

Environment	Class	Keyword	Type	Available
Web	webtable	ColumnCount	Built-In	☑
Web	webtable	CaptureBitmap	Built-In	□
Web	webtable	ChildItem	Built-In	□
Web	webtable	GetROProperty	Built-In	□
Web	webtable	Exist	Built-In	□
Visual Basic	vbtreeview	Drop	Built-In	□
Visual Basic	vblist	Select	Built-In	☑
Visual Basic	vblistview	GetValue	User-Defined	☑
Visual Basic	vbbutton	Click	Built-In	☑
Visual Basic	vbbutton	CaptureBitmap	Built-In	□
Visual Basic	vbbutton	CheckProperty	Built-In	□
Visual Basic	vbbutton	OutputProperty	User-Defined	☑
Visual Basic	vbbutton	ChildObjects	Built-In	□
Visual Basic	vbbutton	SetTOProperty	Built-In	□
Visual Basic	vbbutton	DblClick	Built-In	□

1509 keywords found.

Properties

Description: Navigates to the home page configured in the browser's settings.

Location: Internal

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. **Notes:** Keywords in user-defined functions that are registered to a test object are displayed under the environment and object class to which they are registered. Keywords in user-defined functions that are not registered to a test object, plus built-in VBScript functions, are all displayed under the **Global** environment.
Class	The object class, for example, Image or Winbutton.
Keyword	The displayed operation name, for example, Click or VerifyProperty.
Type	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.

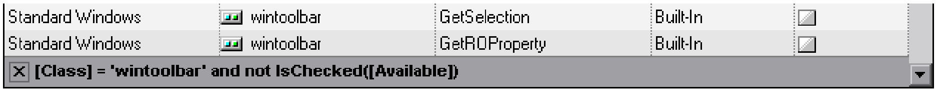
If the data in a column is partially hidden because the column is too narrow, you can resize the column using the mouse. Drag a column header divider to adjust the width.

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.



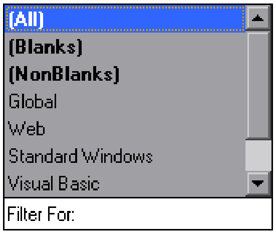
Standard Windows	wintoolbar	GetSelection	Built-In	☐
Standard Windows	wintoolbar	GetROProperty	Built-In	☐

☒ **[Class] = 'wintoolbar' and not IsChecked([Available])**

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

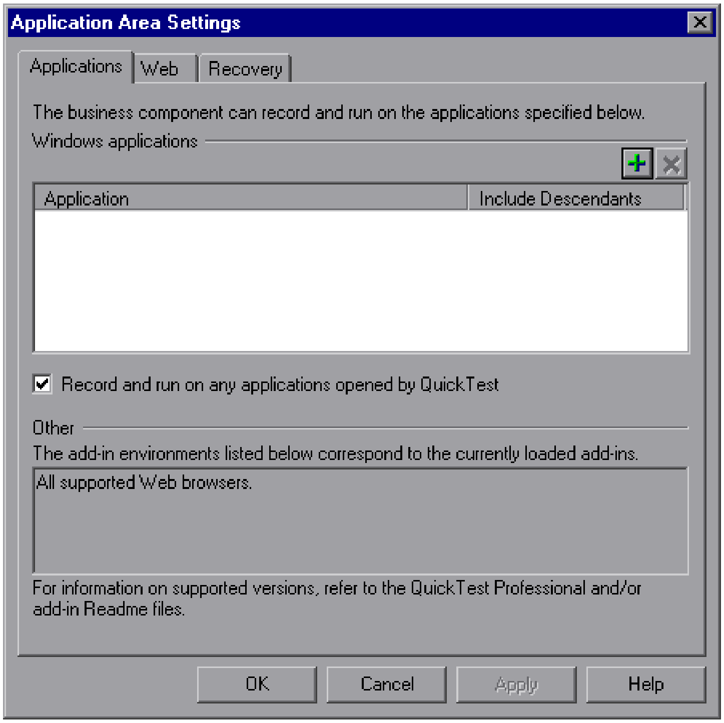
Clicking the **Additional Settings** button in the General pane opens the Application Area Settings dialog box, which comprises several tabs. These tabs enable you to define specific settings for your application area, such as the applications on which the components associated with the application area 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).



Application Area Settings

The business component can record and run on the applications specified below.

Windows applications

Application	Include Descendants

☑ Record and run on any applications opened by QuickTest

Other

The add-in environments listed below correspond to the currently loaded add-ins.

All supported Web browsers.

For information on supported versions, refer to the QuickTest Professional and/or add-in Readme files.

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.

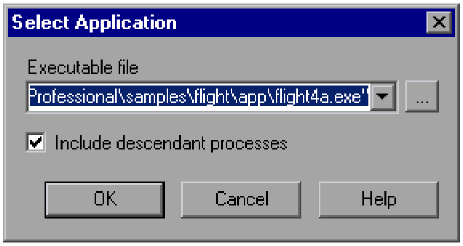
The Applications dialog box and Applications tab may also contain options applicable to any QuickTest external add-ins installed on your computer. For information regarding these options, refer to the documentation provided with the specific add-in.

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 102. 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 102.
✕	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.



Executable file: Professional\samples\flight\app\flight4a.exe"

☑ Include descendant processes

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.

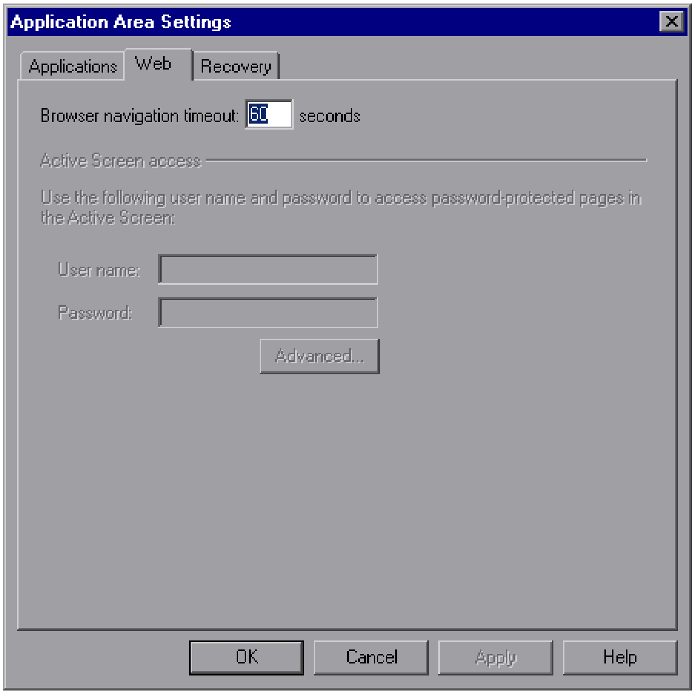
The details entered in the Select Application dialog box are displayed as a single line for each application in the **Windows applications** area of the Applications tab.

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. By default, this option is selected.

Defining Web Settings for Your Application Area

The Web tab provides an option that defines how long to wait for a Web page to load.



Browser navigation timeout: 60 seconds

Use the following user name and password to access password-protected pages in the Active Screen:

User name:

Password:

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

The Web tab includes the following options:

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

Tip: In addition to the options in this tab, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as moving the pointer over an object to open a sub-menu, you may need to modify your Web event configuration to recognize such events. For more information, see Chapter 12, "Configuring Web Event Recording."

Defining Recovery Scenario Settings for Your Application Area

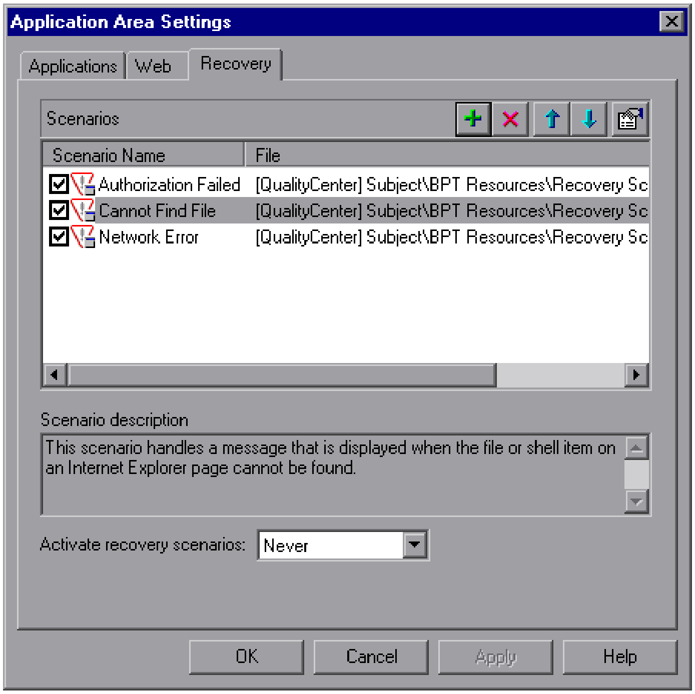
Recovery scenario settings enable you to specify how a business component recovers from unexpected events and errors during a run session.

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 22, "Defining and Using Recovery Scenarios."



Scenario Name	File
☑ Authorization Failed	[QualityCenter] Subject\BPT Resources\Recovery Sc
☑ Cannot Find File	[QualityCenter] Subject\BPT Resources\Recovery Sc
☑ Network Error	[QualityCenter] Subject\BPT Resources\Recovery Sc

Scenario description

This scenario handles a message that is displayed when the file or shell item on an Internet Explorer page cannot be found.

Activate recovery scenarios: Never

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 106.
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: • **On every step.** The recovery mechanism is activated after every step. • **On error.** The recovery mechanism is activated only after steps that return an error return value. • **Never.** The recovery mechanism is disabled. **Note:** Choosing **On every step** may result in slower performance during the run session.

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 22, “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 22, “Defining and Using Recovery Scenarios.”

Saving an Application Area

You can save an application area before or after you define its settings and resources.

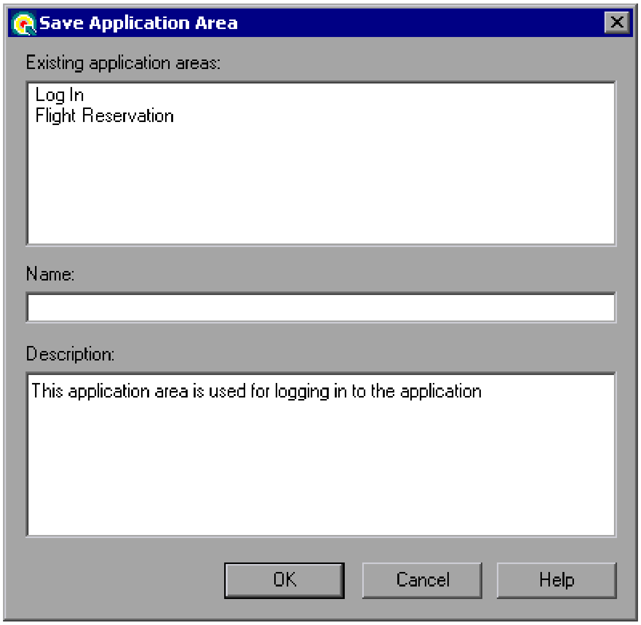
When you save an application area, make sure that you provide a unique name and description that clearly indicate its use. For example, if the application area is intended to be used by components that test the Log In module, you might name it "Log In" and add a description that specifies its intended use, such as, "Intended for use with business components that test the Log In 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 22.

2 Create an application area and modify its settings as required. For more information, see "Creating an Application Area" on page 73.

3 Click **Save** or choose **File** > **Save**. The Save Application Area dialog box opens.



Save Application Area

Existing application areas:

Log In
Flight Reservation

Name:

Description:

This application area is used for logging in to the application

The Save Application Area dialog box includes the following:

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><b>Existing application areas</b></td><td>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.</td></tr>
<tr><td><b>Name</b></td><td>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.
<b>Note:</b> The name you enter cannot exceed 220 characters, cannot contain begin or end with spaces, and cannot contain the following characters:
\ / : " ? < > | * ! { } ‘ % ;</td></tr>
<tr><td><b>Description</b></td><td>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 73.
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.
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.</td></tr>
</table>

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 22.

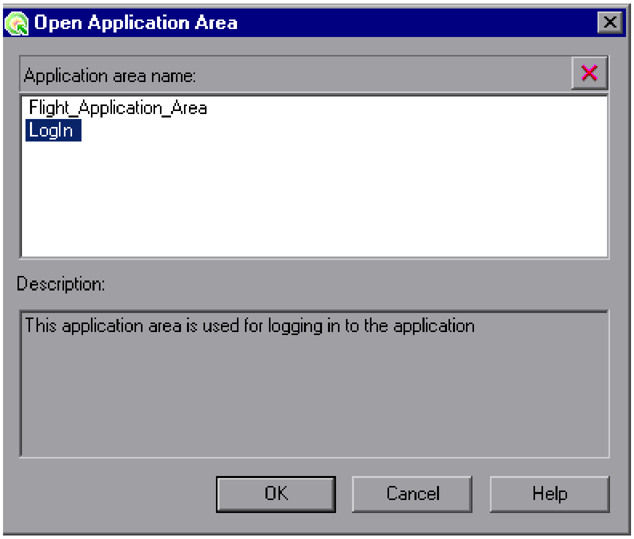
2 In QuickTest, open the application area, (if it is not open).

➤ 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.


Open Application Area

Application area name:

Flight_Application_Area
LogIn

Description:

This application area is used for logging in to the application

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.

4

Working with Business Components

You can use the Business Component Keyword View to create, view, modify, and debug a business component in QuickTest.

About Working with Business Components

Generally, business components are created and modified in Quality Center by Subject Matter Experts. For more information, refer to the *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 647.

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 *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 129 and "Working with Comments" on page 179.

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 *Business Process Testing User's Guide*.

If needed, you can convert a business component to a scripted component. For more information, see Chapter 5, "Working with 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 3, “Working with Application Areas.”

Generally, business components are created in Quality Center by Subject Matter Experts. For more information, refer to the *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 *Mercury Quality Center Administrator's Guide*.

To create a new business component:

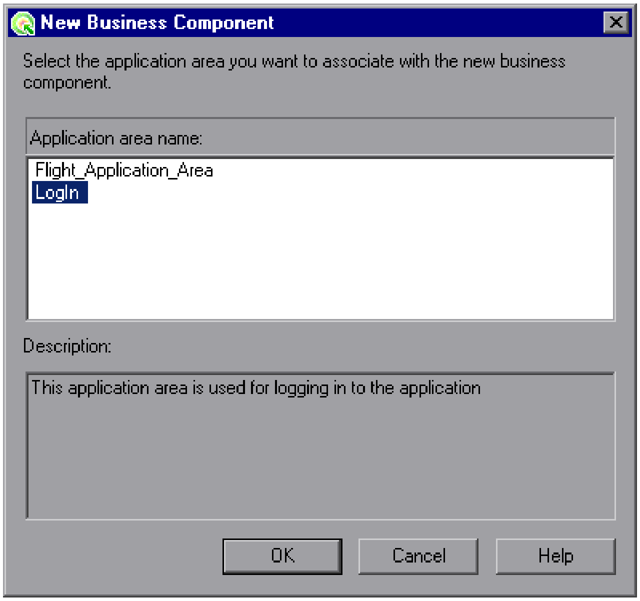
1 Connect to the Quality Center project in which you want to save the business component. For more information, see “Connecting to Your Quality Center Project” on page 22.

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.)



New Business Component

Select the application area you want to associate with the new business component.

Application area name:

Flight_Application_Area
LogIn

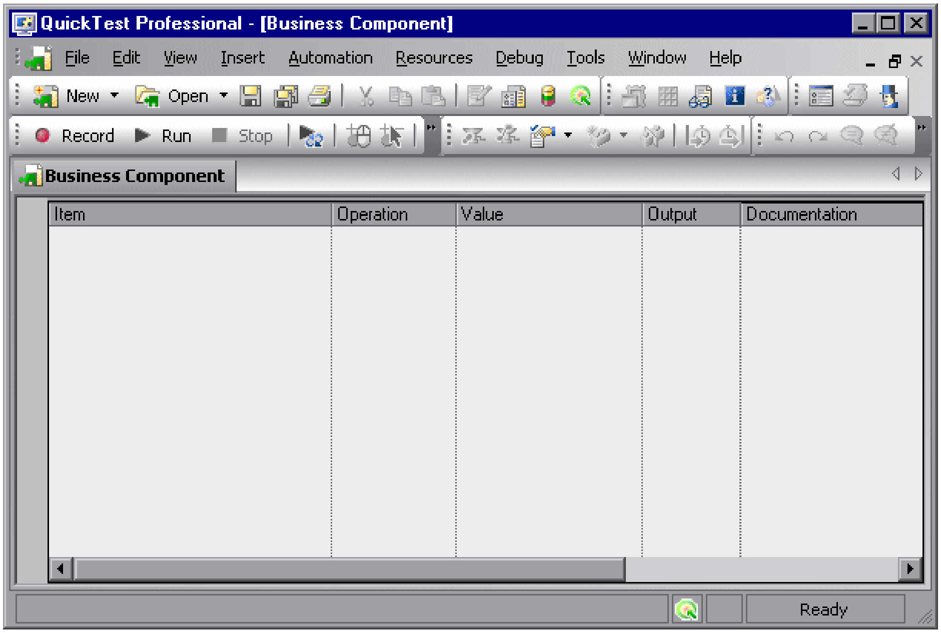
Description:

This application area is used for logging in to the application

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 3, "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.



Item	Operation	Value	Output	Documentation

4 You can now:

- ➤ Add steps and comments to your business component. For more information, see "Adding a Step to Your Component" on page 152 and "Working with Comments" on page 179.
- ➤ Save your component. (You can add steps later.) For more information, see "Saving a Business Component" on page 122.

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 QuickTest 9.0 or earlier, you are asked whether you want to convert it or view it in read-only format. If you choose 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 9.1 on a computer running QuickTest 8.2.

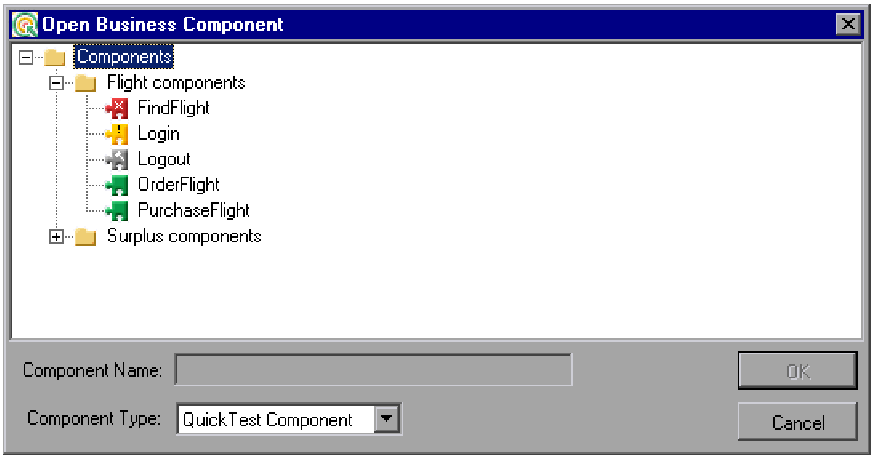
You open business components and scripted components in the same way. For more information on scripted components, see "Working with Scripted Components" on page 135.

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 22.

2 Choose **File** > **Open** > **Business/Scripted Component**, or click the **Open** 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.



Component Name:

Component Type: QuickTest Component

In the component tree, the status of each component is indicated by its icon. For more information, refer to the *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 122.

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 126.
- ➤ **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.

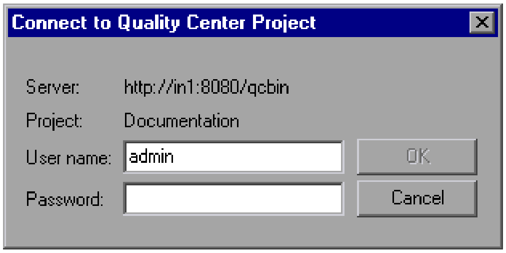
As QuickTest downloads and opens the component, the operations it performs are displayed in the status bar.

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.



Connect to Quality Center Project

Server: http://in1:8080/qcbin

Project: Documentation

User name: admin

Password:

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 "Working with Scripted Components" on page 135.

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 142.

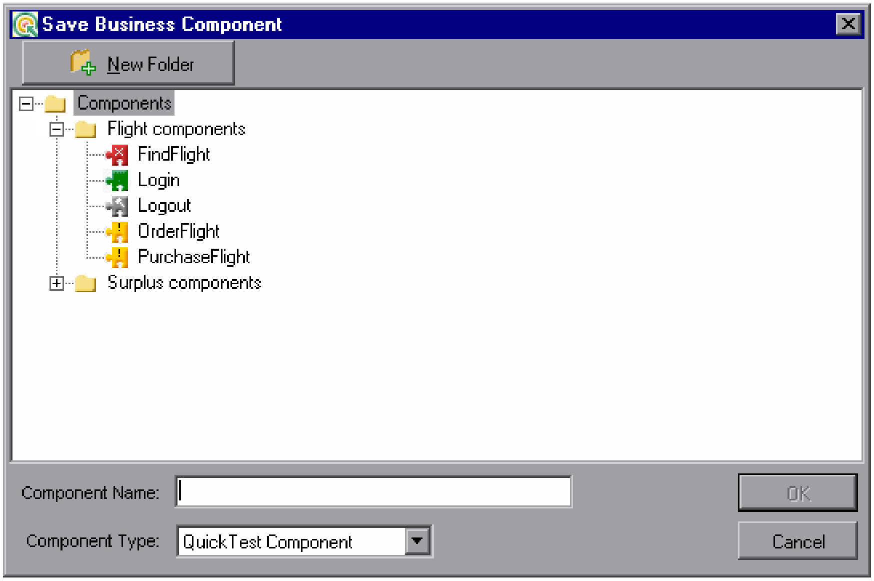
Note: For scripted components only, the data sheet name in the Data Table is identical to the scripted component name. If you save a scripted component with a new name (**File** > **Save As**), the data sheet is automatically renamed. If you have a step that references the data sheet by name, the step 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.



Component Type: QuickTest Component

In the component tree, the status of each component is indicated by its icon. For more information, refer to the *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 *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

You can convert a manual component created in Quality Center to a 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 a 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. You can also modify the manual steps and add additional manual steps, as needed.

Note: You can also convert a manual component to a business component from within Quality Center. For more information, refer to the *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 a 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 22.

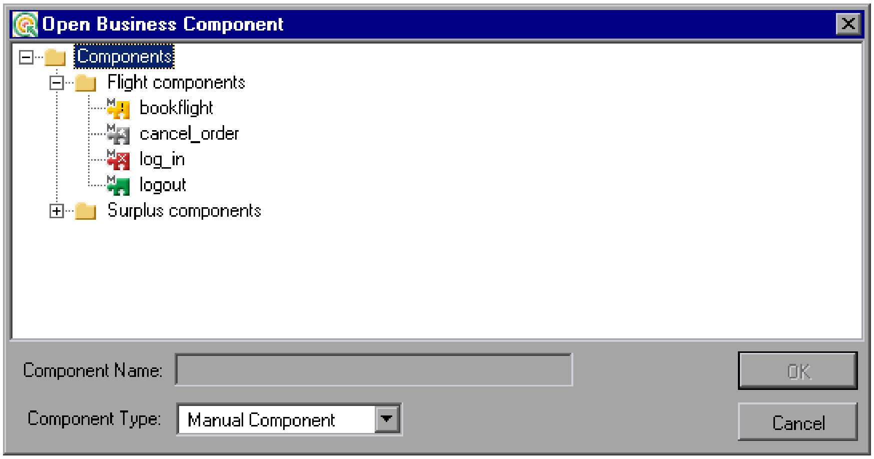
2 Perform one of the following:

➤ Choose **File** > **Open** > **Business/Scripted Component**

➤ Click the **Open** 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.



Component Name:

Component Type: Manual Component

In the component tree, the status of each component is indicated by its icon. For more information, refer to the *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.

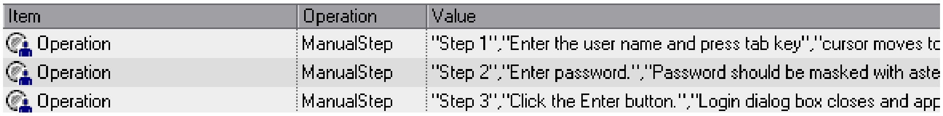
6 Click **OK** to open the component. QuickTest asks whether you want to convert the manual component to a business component.

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 3, “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.



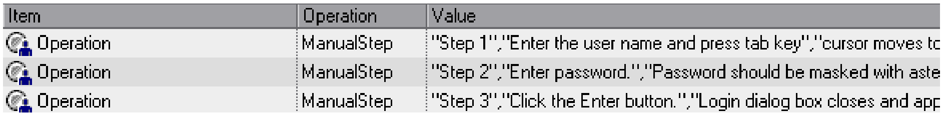
Item	Operation	Value
Operation	ManualStep	"Step 1","Enter the user name and press tab key","cursor moves tc
Operation	ManualStep	"Step 2","Enter password.","Password should be masked with aste
Operation	ManualStep	"Step 3","Click the Enter button.","Login dialog box closes and apr

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," below.

Adding and Modifying Manual Steps for Components

When you convert a manual component to a business 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 tc
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 apr

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 **ManualStep** operations to the component in the Design Steps tab in Quality Center, and 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 145. 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 *Business Process Testing User's Guide*.

Changing the Application Area Associated with a Component

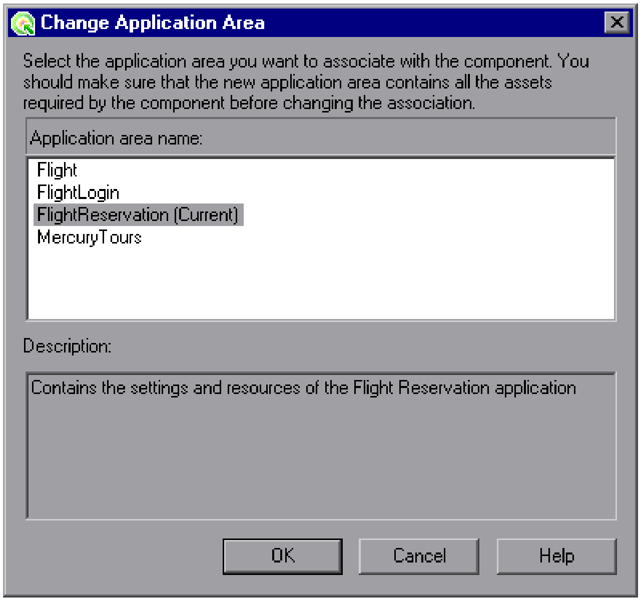
When you create a business component in QuickTest, you must select the application area to which you want to associate the component. There may be one or more application areas available from which to choose. You should select the one that is best suited for the component.

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 7, “Handling Missing Resources.”

To change the application area:

1. Open the component as described in “Opening a Business Component” on page 119.
2. Choose **File** > **Change Application Area**. The Change Application Area dialog box opens.



Select the application area you want to associate with the component. You should make sure that the new application area contains all the assets required by the component before changing the association.

Application area name:

Flight
FlightLogin
FlightReservation (Current)
MercuryTours

Description:

Contains the settings and resources of the Flight Reservation application

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.

5

Working with 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 only describes how to create scripted components.

About Working with 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 component in the same way as you save and open business components. For more information, see “Saving a Business Component” on page 122 and “Opening a Business Component” on page 119.

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 24, “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 automatically at the start of a record or run session.

For general information on all of the functionality available for scripted components, refer to the *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.

Note: The Keywords pane is not relevant for scripted components.

- 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 69.

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 142.

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:

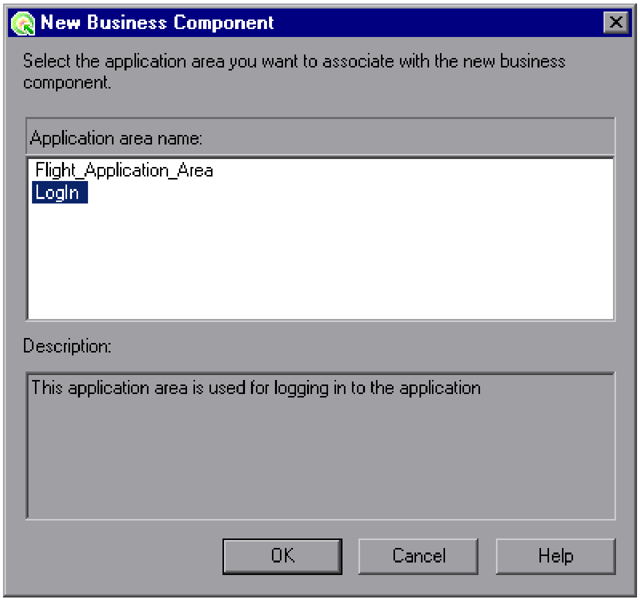
1 Connect to the Quality Center project in which you want to save the scripted component. For more information, see “Connecting to Your Quality Center Project” on page 22.

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.



New Business Component

Select the application area you want to associate with the new business component.

Application area name:

Flight_Application_Area
LogIn

Description:

This application area is used for logging in to the application

---

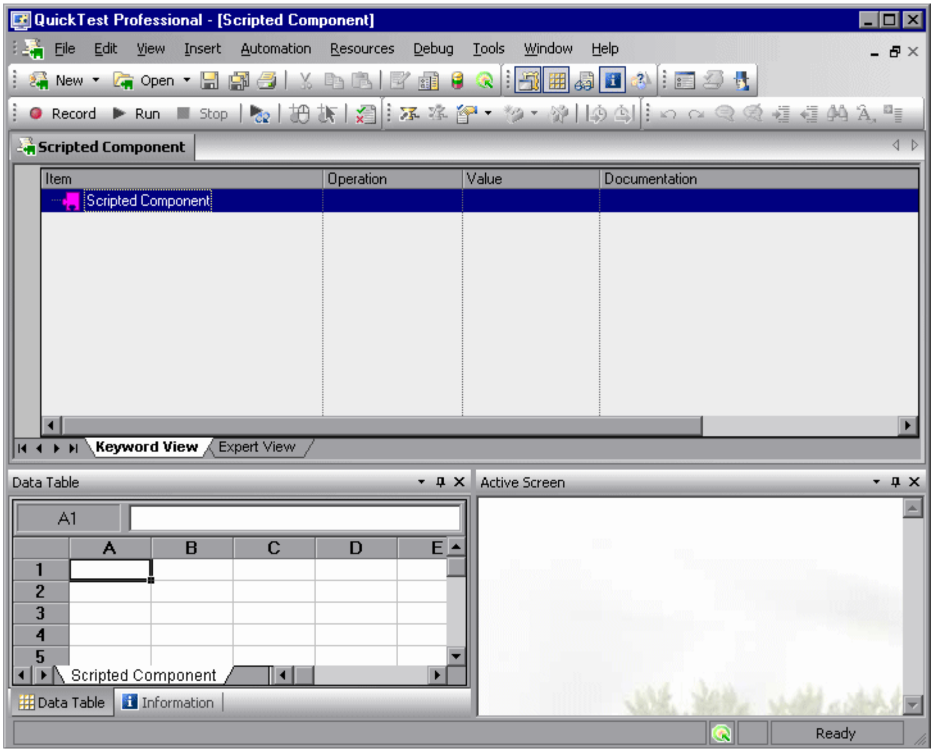
Note: If you have not yet created an application area, a new, untitled scripted component opens using the default settings that are supplied with Business Process Testing. You will not be able to change the application area later.

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 *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 122.

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 *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 119.

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 can use the Action Conversion Tool to convert QuickTest test actions to scripted components for use in Business Process Testing.

To open the Action Conversion Tool, choose **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Action Conversion Tool**.

For information on working with the Action Conversion Tool, click the **Help** button in the Action Conversion Tool window.

6

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.

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.

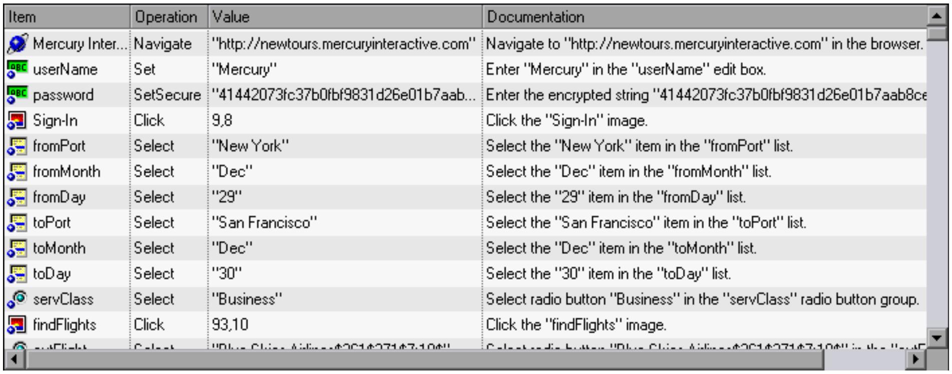
The Business Component Keyword View differs from the QuickTest Test Keyword View in that it provides component-specific options that are specially designed to facilitate the creation of business components. This makes it easy and intuitive for Subject Matter Experts to create business components in Quality 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.

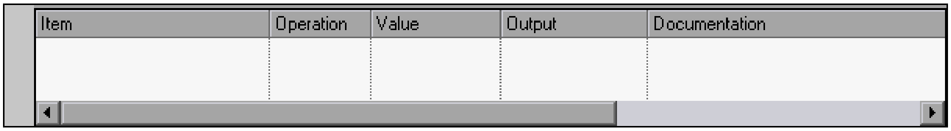
Understanding the Keyword View

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 183.



Item	Operation	Value	Documentation
Mercury Inter...	Navigate	"http://newtours.mercuryinteractive.com"	Navigate to "http://newtours.mercuryinteractive.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.



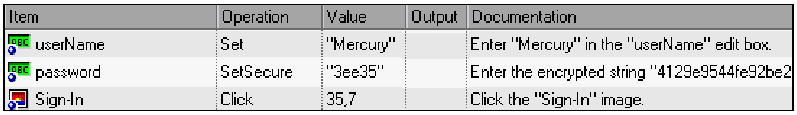
Item	Operation	Value	Output	Documentation

As you add steps to it, 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 a comment.

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 179.

You can add steps to a component manually or by recording the steps you perform on your application. During a recording session, each step is recorded 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 133.

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 154.

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 162. 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 621.

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 170.

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 263.

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: In the Keyword View, if you do not see one or more of these columns, you can use the Keyword View Options dialog box to display them.
For more information, see “Defining Keyword View Display Options” on page 183.

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 Commands Using Shortcut Keys” on page 52.

Adding a Step to Your Component

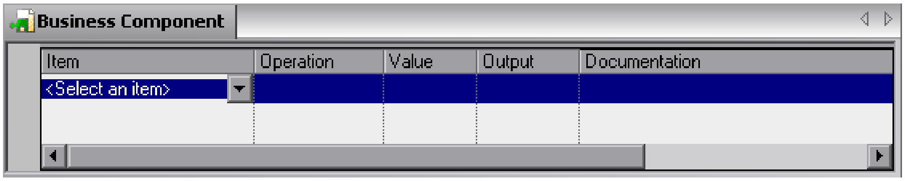
You can add a step at any point in your component. You can add a step below the currently selected step, at the end of a component, or at the beginning of a new component.

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.



Item	Operation	Value	Output	Documentation
<Select an item>				

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 154.

  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 179.

- **Operation.** The operation to be performed on the item. For more information, see "Selecting the Operation for Your Step" on page 162.
- **Value.** (If relevant.) The argument values for the selected operation. For more information, see "Defining Values for Your Step Arguments" on page 164.
- **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 167.

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 647 and "Managing Function Libraries" on page 83.

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 181 and "Using Keyboard Commands in the Keyword View" on page 182.

3 After you make your changes, save the component to your Quality Center project. For more information, see "Saving a Business Component" on page 122.

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 83 and "Working with User-Defined Functions and Function Libraries" on page 621.)

This section describes:

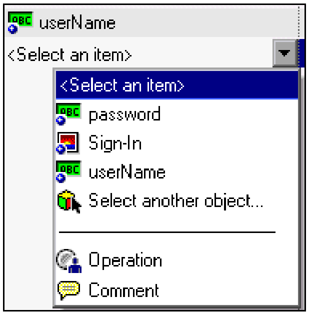
- "Selecting a Test Object from the Item List" on page 155
- "Selecting a Test Object from the Shared Object Repository" on page 156
- "Selecting a Test Object from Your Application" on page 158
- "Selecting the Operation Item" on page 162

After you select an item, you specify an operation for it. For more information, see "Selecting the Operation for Your Step" on page 162.

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 179.

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 162.

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).

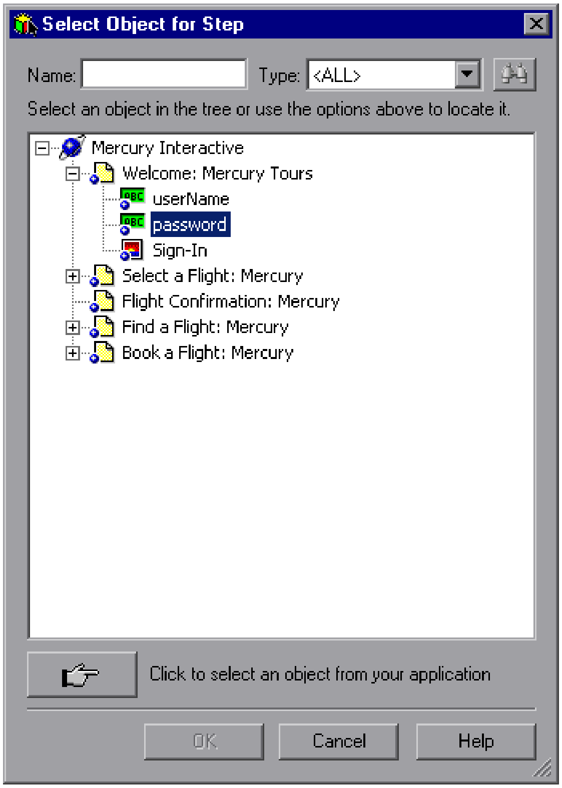
You can select any object in the object repository tree for your new step. If the object repository is very large, you can search for the object. For example, you may want to add a **password** object that you know is an Edit box. You can search all the **Edit** type objects for one called **password**, or even one containing the letter **p**.

For more information on the object repository, see Chapter 14, "Working with Test Objects." For more information on **Object** statements, see "Accessing Run-Time Object Properties and Methods" on page 701.

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 step 8.

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.

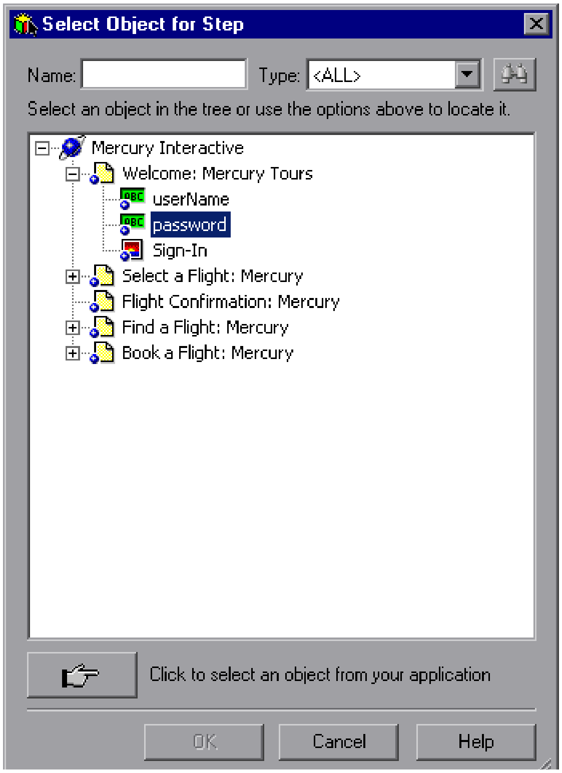
8. Click **OK**. The object is displayed in the **Item** column of the Keyword View. and is also added to the **Item** list. You can now specify the operation for the selected object. For more information, see "Selecting the Operation for Your Step" on page 162.

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 **Select another object**. The Select Object for Step dialog box opens.



Select Object for Step

Name:
Type: <ALL>

Select an object in the tree or use the options above to locate it.

- Mercury Interactive
  - Welcome: Mercury Tours
    - userName
    - password
    - Sign-In
  - Select a Flight: Mercury
  - Flight Confirmation: Mercury
  - Find a Flight: Mercury
  - Book a Flight: Mercury

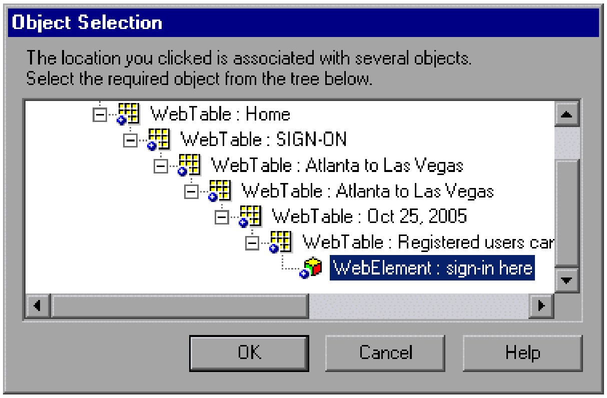
Click to select an object from your application

3. Click the pointing hand button. QuickTest is minimized.

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.



Object Selection

The location you clicked is associated with several objects.
Select the required object from the tree below.

- WebTable : Home
  - WebTable : SIGN-ON
    - WebTable : Atlanta to Las Vegas
      - WebTable : Atlanta to Las Vegas
        - WebTable : Oct 25, 2005
          - WebTable : Registered users car
            - WebElement : sign-in here

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 162.

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 468.

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 346.

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 162.

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 function(s) 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 83.

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 621.

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 647.

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.

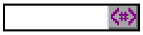
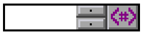
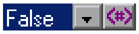
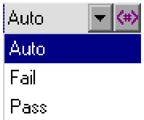
If you selected **Operation** in the **Item** column, QuickTest displays the functions that you defined in the function library, alphabetically. (You manage the function libraries for components in the Function Libraries pane of the Application Area. For more information, see "Managing Function Libraries" on page 83.)

Defining Values for Your Step Arguments

The **Value** cell lists the value(s) for the operation argument(s). You can insert a constant value or a parameter. 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 170 and "Defining Parameters for Your Component" on page 263.

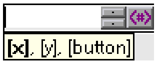
You can also encode password values. For more information, see "Inserting Encoded Passwords into Method Arguments" on page 166.

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 any alphanumeric string 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.
	Predefined Constant	Enables you to select a predefined value from the list (as shown in the example on the left).

To define or modify a value:

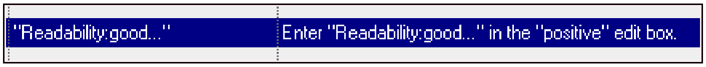
Click in each partition of the **Value** cell and enter the argument values for the selected operation. Note that when you click in the **Value** cell, a tooltip displays information for each argument. In the tooltip, the argument for the partition that is currently highlighted is displayed in bold, and any optional arguments are enclosed in square brackets.



[x], [y], [button]

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.



"Readability:good..."	Enter "Readability:good..." in the "positive" edit box.

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:

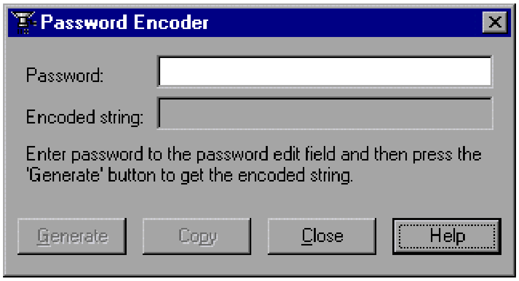
Click the button in the required **Value** cell. For information on parameterizing a local value, see “Working with Parameters” on page 170. For information on parameterizing a component value, see “Defining Parameters for Your Component” on page 263.

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.



Enter password to the password edit field and then press the 'Generate' button to get the encoded string.

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:

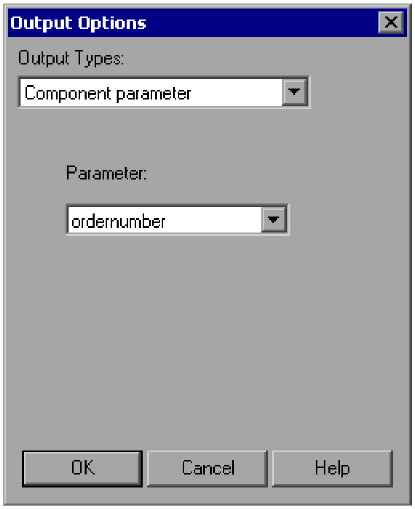
- If at least one output component parameter is defined in the component, the default output type is **Component parameter** and the default output name is the first output parameter displayed in the Parameters tab of the Business Component Settings dialog box.
- If no output component parameter is defined 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 263.

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 169.

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.



Output Options

Output Types: Component parameter

Parameter: ordernumber

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 263.

---

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 170.
- ➤ If you select a local parameter, specify the details for it. For more information, see “Working with Parameters” on page 170.
- ➤ 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.

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 a test or 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 call. 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 152.

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 263.

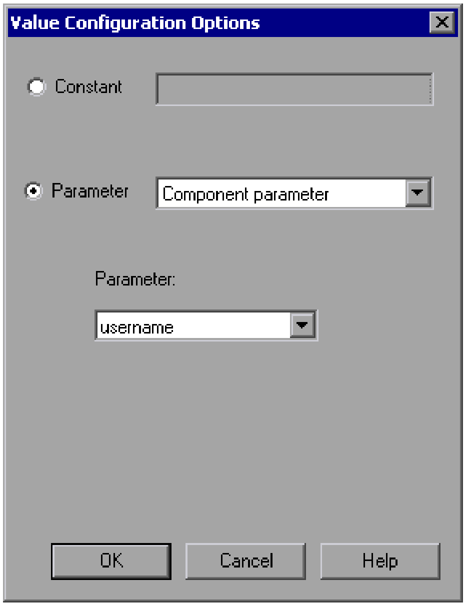
After you define a parameter you can use it to parameterize a value. Alternatively, you can apply a constant value to the parameter by typing it directly in the **Value** cell.

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.



Value Configuration Options

○ Constant

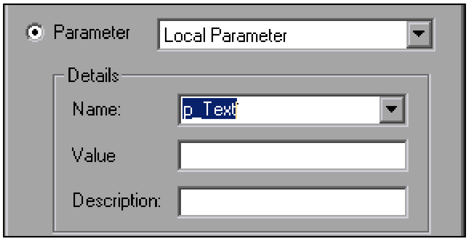
● Parameter: Component parameter

Parameter: username

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.



● Parameter: Local Parameter

Details

Name: p_Text

Value:

Description:

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.

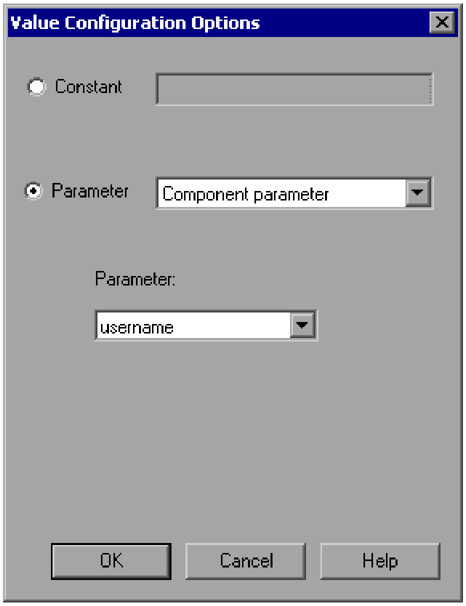
Tips:

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.



Value Configuration Options

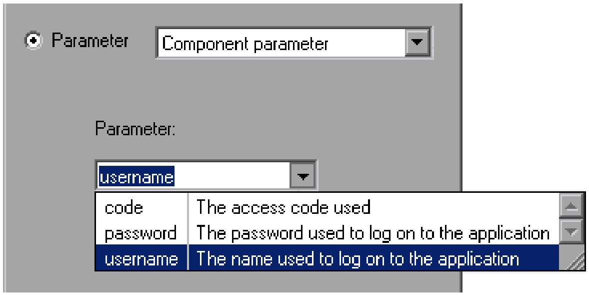
○ Constant

● Parameter: Component parameter

Parameter: username

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 263.

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.



● Parameter: Component parameter

Parameter: username

code	The access code used
password	The password used to log on to the application
username	The name used to log on to the application

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.

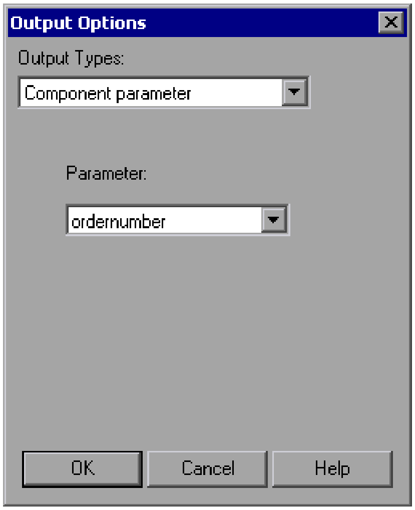
If you click in the **Value** cell after you define a component parameter for it, the icon is displayed in each part of the cell for which a component parameter is defined.

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.



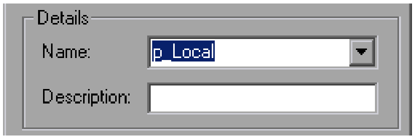
Output Options

Output Types: Component parameter

Parameter: ordernumber

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.



Details

Name: p_Local

Description:

3 Specify the property details for the local parameter:

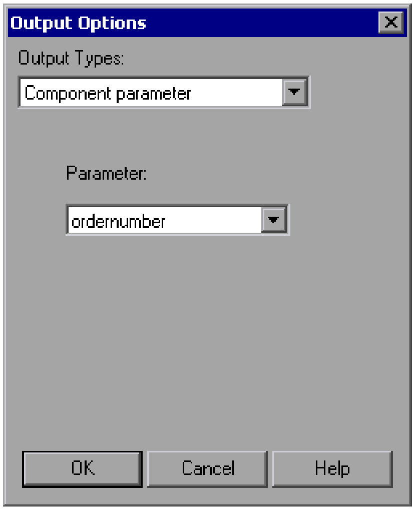
- **Name.** Enter a meaningful name for the parameter or choose one from the list.
- **Description.** Enter a brief description for the parameter.

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.



Output Options

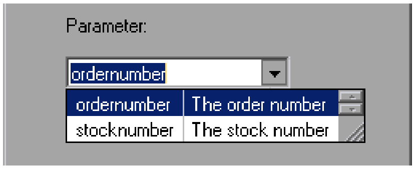
Output Types: Component parameter

Parameter: ordernumber

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 263.

---

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.



Parameter:

ordernumber

ordernumber	The order number
stocknumber	The stock number

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 toimprove 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 183. 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, and **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 letter(s). The typed sequence is highlighted 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.

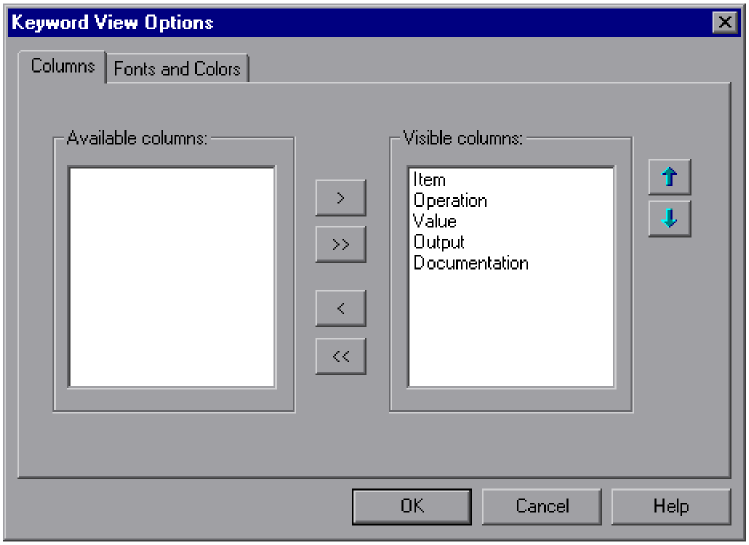
Displaying Keyword View Columns

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 108.

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.

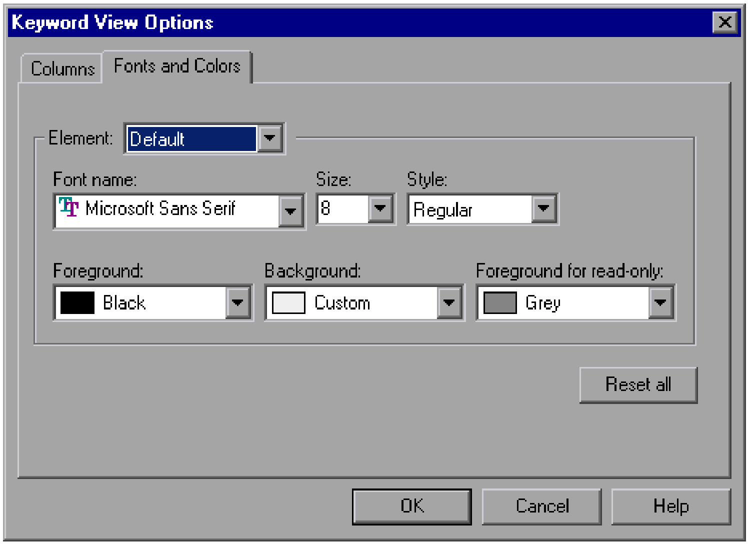
Note: The order of the columns in the Keyword View does not affect the order in which the cells need to be completed for each step. For example, if you choose to display the **Operation** column to the left of the **Item** column, you still need to select the item first, and only then is the **Operation** 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.



Element: Default

Font name: Microsoft Sans Serif

Size: 8

Style: Regular

Foreground: Black

Background: Custom

Foreground for read-only: Grey

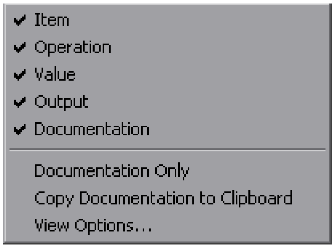
The Fonts and Colors tab includes the following options:

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: • **Alternate Rows**. The background color of every other row. The font and text color for the alternate rows is the same as the font and text color defined for the **Default** element. • **Comment**. The row and text of comment lines. Note that all of the available formatting options apply to entire comment rows, not to comments within a regular step row. For comments within a step row, only the specified **Foreground** color applies (all other settings are taken from the **Alternate Rows**, **Default**, or **Selected Row** settings, as appropriate). • **Default**. All rows and text in the Keyword View (except for the elements listed below). • **Selected Row**. The row and text currently selected (highlighted).
Font Name	Enables you to modify the font used for text in the selected element. You cannot change the font for **Alternate Rows** or **Selected Row** elements. **Note:** When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your 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.

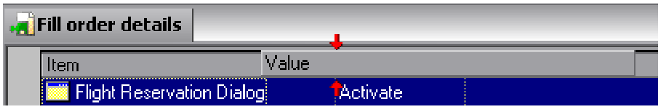
Tips for Working with the Keyword View

- 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 quickly 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 easily 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 516.

7

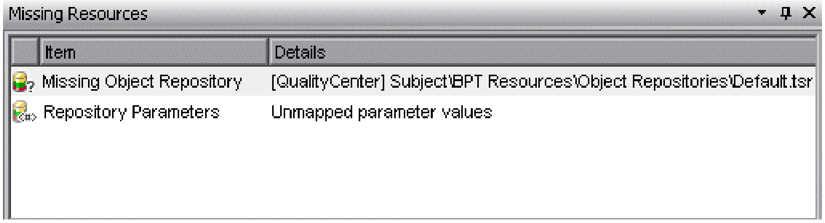
Handling Missing Resources

If a component's 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's application area, as required.

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. Specifically, QuickTest verifies that all associated shared object repositories can be found and that all defined repository parameters contain values.

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, enabling you to remap or remove them from your component or application area. After you successfully handle a missing resource, QuickTest removes it from the pane.



Missing Resources

Item	Details
Missing Object Repository	[QualityCenter] Subject\BPT Resources\Object Repositories\Default.tsr
Repository Parameters	Unmapped parameter values

The Missing Resources pane may list any of the following types of missing resources:

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 194.

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 195.

Filtering the Missing Resources Pane

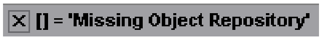
You can choose to display all missing resources in the Missing Resources pane, or only one type of missing resource.

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.
- **Missing Object Repository.** Displays a row for each shared object repository that cannot be found, specifying the path QuickTest uses to search for the shared object repository.
- **Object Repository Parameter.** Displays a generic row indicating that at least one test object in the repository has at least one parameterized 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:

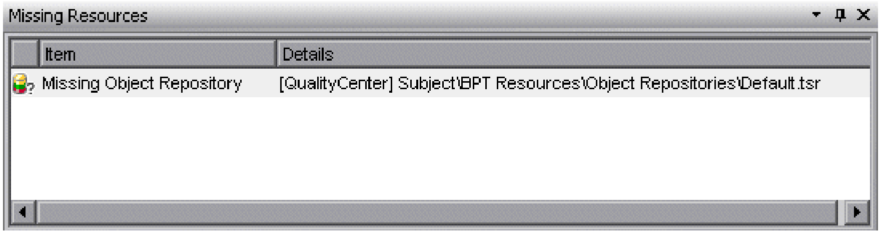


☒ [] = 'Missing Object Repository'

Tip: You can cancel the filter and show all missing resources again by clicking the ☒ icon on the left of the filter indication.

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.



Missing Resources

Item	Details
Missing Object Repository	[QualityCenter] Subject\BPT Resources\Object Repositories\Default.tsr

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.

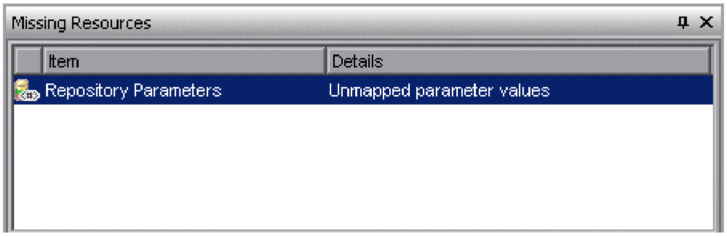
For a component, if you double-click the line displaying the missing object repository, QuickTest displays a message explaining that the object repository must be mapped to the associated application area. You or the Automation Engineer needs to open the application area and correct the association of 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 88.

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 433.)

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.



Item	Details
Repository Parameters	Unmapped parameter values

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 double-click the line displaying **Repository Parameters**, the Map Shared 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 357.

28

Selecting QuickTest Add-Ins

QuickTest Professional supports testing on standard Windows applications. It has several built-in add-ins, including Web, ActiveX, and Visual Basic, that can be installed from the QuickTest Professional setup. Installing and loading these add-ins enables QuickTest to work with the corresponding environments. You can also install additional external add-ins to meet your testing needs.

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

About Working with QuickTest Add-Ins

You can install the QuickTest built-in add-ins (Web, ActiveX, Visual Basic) when you install QuickTest Professional, or you can install the QuickTest built-in add-ins at a later time by running the installation again. Add-ins that are installed separately from the QuickTest Professional installation are referred to as **external** add-ins.

Each external add-in requires a seat or concurrent license code. You install a seat add-in license on your computer using the Add-in Manager dialog box. You install a concurrent add-in license on the Mercury Functional Testing Concurrent License Server computer.

For more information on installing add-ins and licenses, refer to the *QuickTest Professional Installation Guide*.

About QuickTest Built-in Add-ins

ActiveX Add-in. You can use the built-in ActiveX Add-in to test ActiveX controls. You can create and run components on these controls, as well as check their properties. You create and run components on ActiveX controls in much the same way as you do for standard Windows applications.

For information on supported ActiveX controls and versions, refer to the *QuickTest Professional Readme*.

Visual Basic Add-in. You can use the built-in Visual Basic Add-in to test Visual Basic applications. You can create and run components on these controls, as well as check their properties. You create and run components on Visual Basic applications in much the same way as you do for standard Windows applications.

To test Visual Basic .NET Windows Forms applications, you must install and load the .NET Add-in. Note that the .NET Add-in is an external add-in and is not included with your core QuickTest installation. For more information on this add-in, contact your QuickTest supplier or Mercury Customer Support.

Web Add-in. You can use the built-in Web Add-in to test your Web pages and applications. You can test Web objects such as hyperlinks, images, image maps, and Viewlink objects. For more information on using the Web Add-in, see "Working with the Web Add-in" on page 204.

When QuickTest opens, you can choose which of the installed add-ins you want to load using the QuickTest Professional - Add-In Manager dialog box.

If you choose to load an installed add-in, QuickTest recognizes the objects you record on the corresponding environment and enables you to work with the appropriate methods, properties, and specialized options.

Loading QuickTest Add-ins

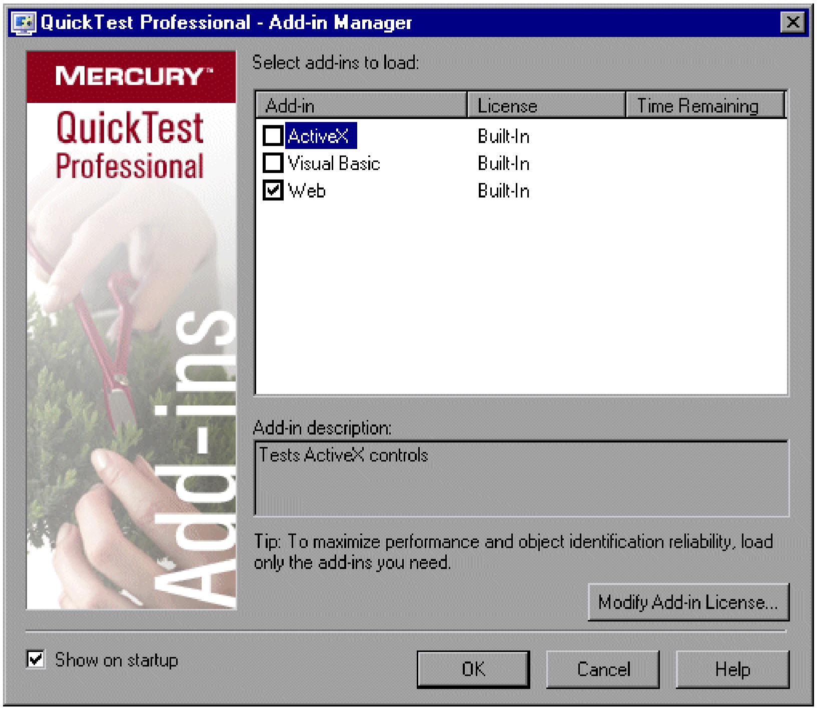
If you have installed QuickTest add-ins, you can specify which add-ins to load at the beginning of each QuickTest session. Loading the relevant add-in enables QuickTest to work with the corresponding environment.

You can use the latest released version of all QuickTest add-ins with QuickTest Professional 9.1. The PeopleSoft Add-in 8.2, Siebel Add-in 8.0, and Terminal Emulators Add-in 8.0 all require you to install the relevant Add-in Upgrade patch before you can use them with QuickTest Professional 9.1.

You can install Add-in Upgrade patches by running the relevant patch executable from the **<QuickTest Professional>\AddinsUpgrade** folder.

Note: If you do not install the relevant Add-in Upgrade patch for these add-ins, they cannot be loaded, and will be labeled as an **Incompatible version** in the Add-in Manager dialog box.

When you start QuickTest, the Add-in Manager dialog box opens. It displays a list of all installed add-ins and the license used for each add-in. If you are using a seat add-in license, it also displays the time remaining for time-limited licenses. For information on the details shown in the Add-in Manager dialog box, see "Understanding the Add-in Manager" on page 201.



Select add-ins to load:

Add-in	License	Time Remaining
□ ActiveX	Built-In	
□ Visual Basic	Built-In	
☑ Web	Built-In	

Add-in description:

Tests ActiveX controls

Tip: To maximize performance and object identification reliability, load only the add-ins you need.

☑ Show on startup

Note: 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.

You can select which add-ins to load for the current session of QuickTest. To maximize performance and object identification reliability, load only the add-ins you need.

Understanding the Add-in Manager

The Add-in Manager dialog box includes the following columns and details:

- **Add-in.** Lists the names of the installed QuickTest Professional add-ins.
- **License.** Lists the license used by the add-in:
  - **Built-In.** Applies to the add-ins that are provided with QuickTest Professional. Built-in add-ins use the same license as QuickTest Professional. Therefore, if QuickTest uses a **Permanent** license, the built-in add-ins use the same **Permanent** license.
  - **Time-Limited.** Temporary license, for example, a demo (14-day) license or a one-year license. (Displayed only when using a QuickTest seat license—not a concurrent license.)
  - **Permanent.** Unlimited expiration date.
  - **Not Licensed.** Applies to an add-in that does not have an installed seat license or access to a concurrent license (for example, if all concurrent licenses are currently in use). To load the add-in, you first need to install or access a license. For more information, refer to the *QuickTest Professional Installation Guide.*
  - **Incompatible Version.** Applies to an add-in that is no longer supported by QuickTest. If a supported (later) version of the add-in is available, you can upgrade to that version or install an upgrade patch, and then load the add-in. For more information, refer to the *QuickTest Professional Installation Guide*.
- **Time Remaining.** Specifies the number of days and hours remaining until a **Time-Limited** add-in license expires. (Displayed only when using a QuickTest seat license—not a concurrent license.)
- **Add-in Description.** Describes the environment that the selected add-in supports.
- **Show on startup.** Instructs QuickTest to display the Add-in Manager dialog box each time you open QuickTest. When this check box is cleared, QuickTest opens and loads the same add-ins it loaded in the previous session, without displaying the Add-in Manager. To display the Add-in Manager again, choose **Tools** > **Options** > **General** and select **Display Add-in Manager on startup**. For information on working with the Options dialog box, see Chapter 25, “Setting Global Testing Options.”

Selecting Add-ins to Load

You select the add-ins that you want QuickTest to load by selecting the check boxes adjacent to required add-ins. When you click **OK**, QuickTest loads the selected add-ins. QuickTest also remembers which add-ins you selected so that the next time you open QuickTest, the same add-ins are selected in the Add-in Manager dialog box.

Matching Loaded Add-ins with Associated Add-ins

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

- close and reopen QuickTest, and select the required add-ins in the Add-in Manager dialog box.
- remove the add-ins from the list of associated add-ins for your component. To change the list of add-ins associated with your component, choose **File** > **Settings** and click **Modify** in the Properties tab. For more information on associating add-ins, see "Associating Add-ins with Your Component" on page 80.

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

- close and reopen QuickTest, and clear the check boxes for the add-ins in the Add-in Manager dialog box, if they are not required.
- add the add-ins to the list of associated add-ins for your component's application area. To change the list of add-ins associated with your component, choose **File** > **Settings** and click **Modify** in the Properties tab. For more information on associating add-ins, see "Associating Add-ins with Your Component" on page 80.

Tips for Working with QuickTest Add-ins

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

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

- You must install and load an add-in to enable QuickTest to recognize objects from the corresponding environment. To load an add-in, select the add-in from the Add-in Manager dialog box that opens when you start QuickTest.
- If the Add-in Manager does not open when you start QuickTest, click the **Options** button or choose **Tools** > **Options** and click the **General** tab. Select the **Display Add-in Manager on startup** check box and click **OK**. Restart QuickTest.
- To maximize performance and object identification reliability, load only the add-ins you need. For example, if you want to test a process that spans a Web application and a .NET application, load only the Web and .NET Add-ins. Do not load all add-ins unless you need to work with all of them.
- You can view the list of add-ins that are currently installed or loaded by choosing **Help** > **About QuickTest Professional**. The dialog box displays a list of all add-ins installed on your computer. A check mark indicates that the add-in is currently loaded.
- If an add-in license has not yet been installed for a specific external add-in, the add-in is displayed as **Not Licensed** in the **License** column of the Add-in Manager dialog box. An add-in may also be displayed as **Not Licensed** if no concurrent license server within your subnet has a registered license for the specific add-in, or if all concurrent licenses are in use (and are, therefore, unavailable). In this case, you can use the LSFORCEHOST variable to connect to a concurrent license server outside of the subnet that has the relevant add-in license installed on it, if one is available. For more information on connecting to concurrent license servers, refer to the *QuickTest Professional Installation Guide*.

Working with the Web Add-in

You can use the Web Add-in to test your Web pages and applications. You can test Web objects such as hyperlinks, images, image maps, and Viewlink objects.

Before you begin recording on Web sites and applications, you should ensure that you have installed and loaded the Web add-in. You can check whether the Web add-in is installed by choosing **Help** > **About QuickTest Professional**. Loaded add-ins are indicated by a check mark in the add-ins list.

You should also set your preferences in the Web tab of the Business Component Settings dialog box, and the Web tab of the Options dialog box. For more information, see Chapter 10, "Working with Business Component Settings" and Chapter 9, "Setting Global Testing Options."

If QuickTest does not record Web events in a way that matches your needs, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as moving the pointer over an object to open a sub-menu, you may need to modify the Web event configuration to recognize such events. For more information, see Chapter 12, "Configuring Web Event Recording."

Note: If you are recording on a list in a Web page or application, you must click on the list, scroll to an entry that was not originally showing, and select it. If you want to select the item in the list that is already displayed, you must first select another item in the list (click it), then return to the originally displayed item and select it (click it). This is because QuickTest only records a step if the value in the list changes.

Working with Web Browsers

You use a Web browser to record components that check Web objects. Before you begin recording, you must open your browser to the relevant Web page.

Note: By default, the name assigned to the Browser test object in the object repository is always the name assigned to the first page recorded for the browser object. The same test object is used each time you record on a browser with the same ordinal ID in future recording sessions. Therefore, the name used for the browser in the steps you record may not reflect the actual browser name.

QuickTest supports recording components on Microsoft Internet Explorer and running components on the following Web browsers:

- Microsoft Internet Explorer
- Netscape Browser
- Mozilla Firefox
- Applications with embedded Microsoft Internet Explorer Web browser controls

Note: QuickTest components are generally cross-browser—you can record a component on Microsoft Internet Explorer and run it on any other supported browser. For information on supported browser versions, refer to the *QuickTest Professional Readme* file.

Working with Microsoft Internet Explorer

Consider the following when using Microsoft Internet Explorer as your Web browser:

- QuickTest Professional Web support behaves as a browser extension in Microsoft Internet Explorer. Therefore, you cannot use the Web Add-in on Microsoft Internet Explorer without enabling the **Enable third-party browser extensions** option. To set the option, in Microsoft Internet Explorer choose **Tools** > **Internet Options** > **Advanced** and select the **Enable third-party browser extensions** option.
- QuickTest Professional does not support tabbed browsing. Therefore, before using the Web Add-in, you must disable tabbed browsing on Internet Explorer. To disable the option, in Microsoft Internet Explorer choose **Tools** > **Internet Options** > **General** > **Change how webpages are displayed in tabs** > **Settings** and clear the **Enable Tabbed Browsing** option. After clearing this option, you must restart your browser before using the QuickTest Professional Web Add-in.

Working with Netscape Browser and Mozilla Firefox

Keep the following in mind when using Netscape Browser or Mozilla Firefox as your Web browser:

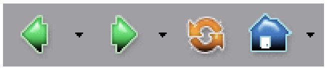
- You must be logged-in with Administrator privileges (or have write permissions to the browser’s installation folder) on the QuickTest computer when launching Mozilla Firefox with QuickTest for the first time, since adding QuickTest support for Mozilla Firefox requires a file to be created in the browser's installation folder.
- You can record components on Microsoft Internet Explorer and run them on Netscape Browser and Mozilla Firefox. You cannot record components on Netscape Browser and Mozilla Firefox. There are two ways to create components to run on Netscape Browser or Mozilla Firefox:
    - Record the component on Microsoft Internet Explorer.
    - Use the keyword-driven methodology: create an object repository for your application using the Object Repository window (local object repository) or Object Repository Manager (shared object repository), and then add steps using the Keyword View or Step Generator.

When you use the keyword-driven methodology, you can add objects using Mozilla Firefox or Netscape Browser if you want; you do not have to use Microsoft Internet Explorer.

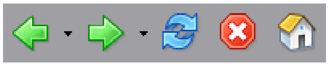
- Generally, components that were recorded on Microsoft Internet Explorer will run on Netscape Browser and Mozilla Firefox without requiring any modification. However, there are several differences that you should keep in mind:

  - QuickTest does not support Netscape Browser and Mozilla Firefox menus or sidebars. The only toolbar buttons that are supported are the **Home**, **Refresh**, **Back**, **Forward**, and **Stop** buttons. All other toolbars and toolbar buttons are not supported.

    The following toolbar buttons are the only supported toolbar buttons for Netscape Browser.

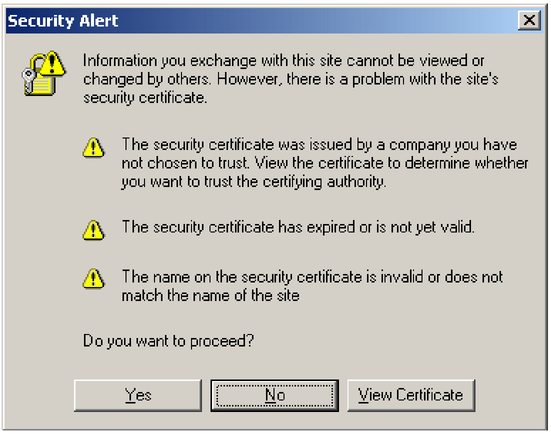
    The following toolbar buttons are the only supported toolbar buttons for Mozilla Firefox.

    If you record steps on any unsupported menu or toolbar objects when working with Microsoft Internet Explorer, you need to remove or replace the steps before running the component on Netscape Browser and Mozilla Firefox.

  - Netscape Browser and Mozilla Firefox use different standard dialog boxes than the Windows standard dialog boxes used by Microsoft Internet Explorer. If your component contains steps on such dialog boxes, you should create appropriate steps to be used when running on Netscape Browser and Mozilla Firefox.

For example, the following two dialog boxes are a security alert of the same Web site, the one on the left is from Microsoft Internet Explorer and the one on the right is from Mozilla Firefox. Although they both look like a Windows dialog box, the Mozilla Firefox one is actually a browser window.



Security Alert

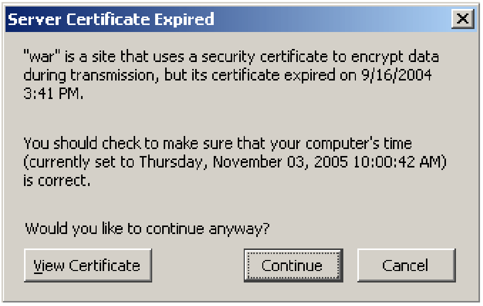
Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

⚠ The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

⚠ The security certificate has expired or is not yet valid.

⚠ The name on the security certificate is invalid or does not match the name of the site

Do you want to proceed?



Server Certificate Expired

"war" is a site that uses a security certificate to encrypt data during transmission, but its certificate expired on 9/16/2004 3:41 PM.

You should check to make sure that your computer's time (currently set to Thursday, November 03, 2005 10:00:42 AM) is correct.

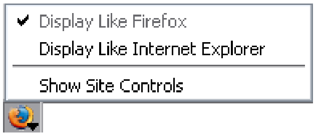
Would you like to continue anyway?

- Due to the difference in standard dialog boxes described above, pop-up recovery scenarios that use the **Click button with label** recovery operation and were built for Microsoft Internet Explorer will not work for Netscape Browser and Mozilla Firefox.

- Tabbed browsing is not supported by QuickTest for any browser type. In Netscape Browser and Mozilla Firefox, you must specifically configure your browser to open a new browser window instead of a new tab. This configuration is dependent on the nature of your application and therefore configuration instructions are not provided. In most cases, you can add objects to the object repository and run components on the active browser tab, even if there are additional tabs currently open.

- Although Netscape Browser supports both Mozilla Firefox and Internet Explorer engines, QuickTest supports only the Mozilla Firefox engine. If your component fails or QuickTest does not recognize any objects in your Web page, make sure Netscape Browser is not using the Internet Explorer engine to display it.

To change the rendering engine to Mozilla Firefox, click the icon in the bottom-left corner of Netscape Browser and select **Display Like Firefox**.



Tips:

You can configure the browser to always display your pages using the Mozilla Firefox engine. (In Netscape Browser, choose **Tools** > **Options** > **Site Controls**, and, in the Site List tab, make sure that the **Firefox** radio button is selected in the **Rendering Engine** area.)

You can configure the browser to always display your pages using the Mozilla Firefox engine by clearing the **Automatically use the Internet Explorer Engine** check box while installing Netscape Browser.

- The **Object** property accesses DOM objects. These are not supported by Netscape Browser and Mozilla Firefox. For more information on the **Object** property, refer to the *QuickTest Professional User's Guide*.

Working with Applications with Embedded Web Browser Controls

To record and run components on an application with embedded Web browser controls:

- Make sure the ActiveX Add-in is loaded.
- Make sure the application is opened after QuickTest.
- Start recording or running the test.

Note: Embedded browser controls are supported only for Microsoft Internet Explorer.

Part III

Configuring Settings

9

Setting Global Testing Options

You can control how QuickTest records and runs components by setting global testing options.

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 Welcome screen 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.

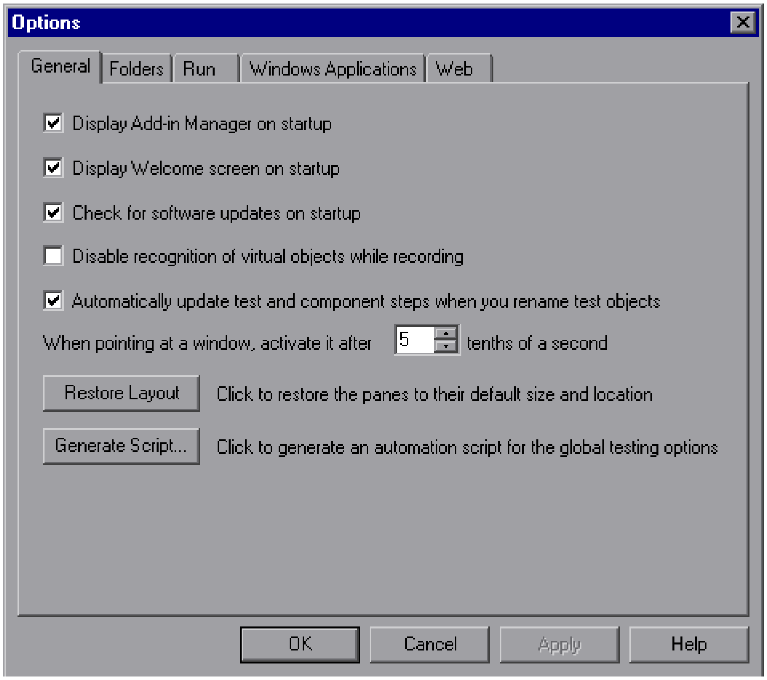
You can also set testing options that affect only the component currently open in QuickTest. For more information, see Chapter 10, “Working with Business Component Settings.”

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 record and run 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.



☑ Display Add-in Manager on startup

☑ Display Welcome screen on startup

☑ Check for software updates on startup

□ Disable recognition of virtual objects while recording

☑ Automatically update test and component steps when you rename test objects

When pointing at a window, activate it after 5 tenths of a second

Click to restore the panes to their default size and location

Click to generate an automation script for the global testing options

---

Note: The Web tab shown above is displayed only if the Web Add-in is installed and loaded.

---

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.

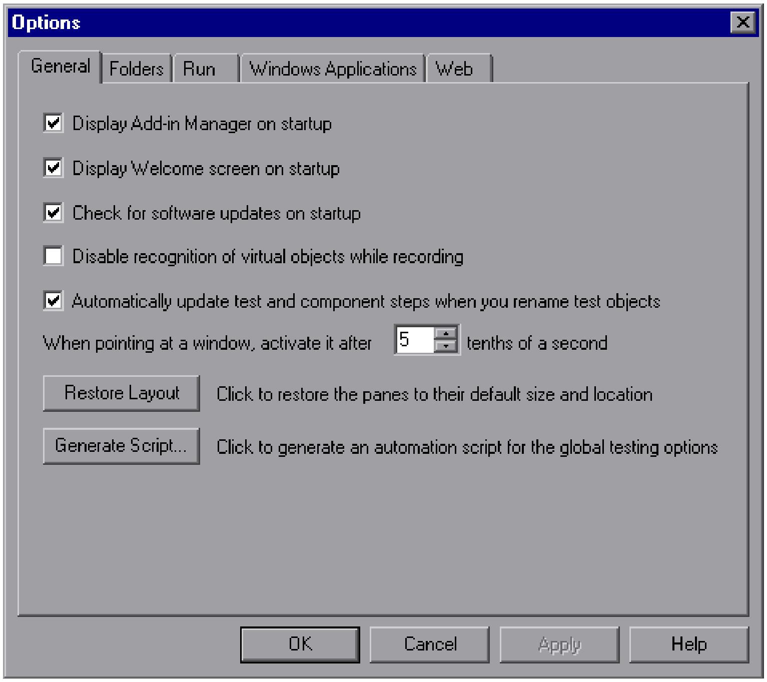
The Options dialog box can contain the following tabbed pages:

Tab Heading	Contains:
General	Options for general component settings. For more information, see "Setting General Testing Options" on page 216.
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 218.
Run	Options for running components. For more information, see "Setting Run Testing Options" on page 220.
Windows Applications	Options for configuring how QuickTest records and runs components for the following Windows applications: • Standard Windows applications • .NET Windows Forms • Visual Basic • ActiveX For more information, see "Setting Windows Application Testing Options" on page 222.
Web (displayed only if the Web Add-in is installed and loaded)	Options for configuring recording and run session behavior in the Web environment. For more information, see "Setting Web Testing Options" on page 235.

The Options dialog box may contain additional tabs for any external add-ins that are currently installed and loaded.

Setting General Testing Options

The General tab options affect the general appearance of QuickTest and other general testing options.



Options

☑ Display Add-in Manager on startup

☑ Display Welcome screen on startup

☑ Check for software updates on startup

□ Disable recognition of virtual objects while recording

☑ Automatically update test and component steps when you rename test objects

When pointing at a window, activate it after 5 tenths of a second

Restore Layout — Click to restore the panes to their default size and location

Generate Script... — Click to generate an automation script for the global 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 "Loading QuickTest Add-ins" on page 199.
Display Welcome screen on startup	Determines whether the Welcome screen is displayed when starting QuickTest.

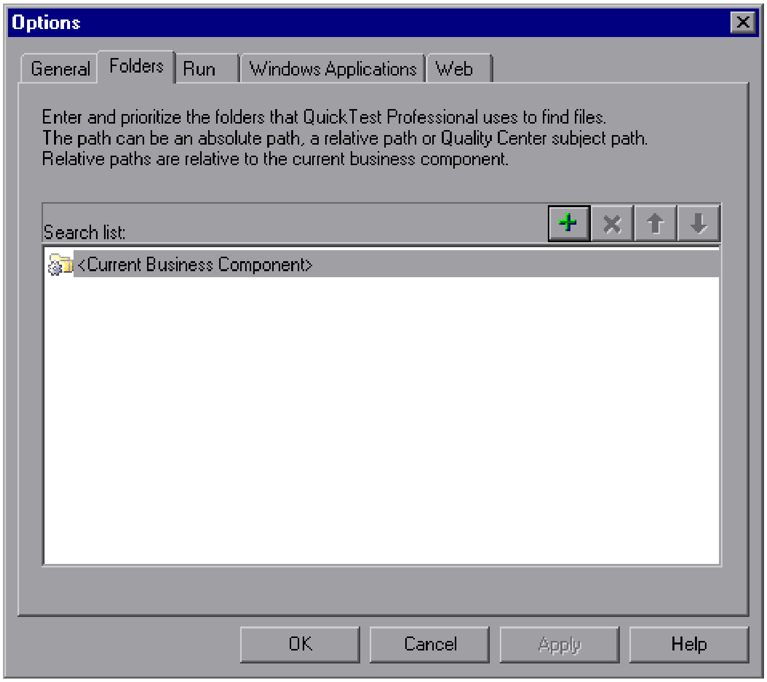
Option	Description
Check for software updates on startup	Instructs QuickTest to automatically check for software updates each time it starts up. For more information, refer to the *QuickTest Professional User's Guide*.
Disable recognition of virtual objects while recording	Determines whether the defined 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 346.
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.**
Restore Layout	Restores the layout of the QuickTest window so that it displays the panes and toolbars in their default sizes and positions.
Generate Script	Generates an automation script containing the current global testing options. For more information, see "Automating QuickTest Operations" on page 709, or refer to the *QuickTest Automation Reference* (**Help** > **QuickTest Professional Help** > **QuickTest Advanced References** > **QuickTest Automation**).

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.

Note: The current component is listed in the Search list by default. It cannot be deleted.

The order in which the folders are displayed in the search list determines the order in which QuickTest searches for the specified component or file.



Enter and prioritize the folders that QuickTest Professional uses to find files.
The path can be an absolute path, a relative path or Quality Center subject path.
Relative paths are relative to the current business component.

Search list:

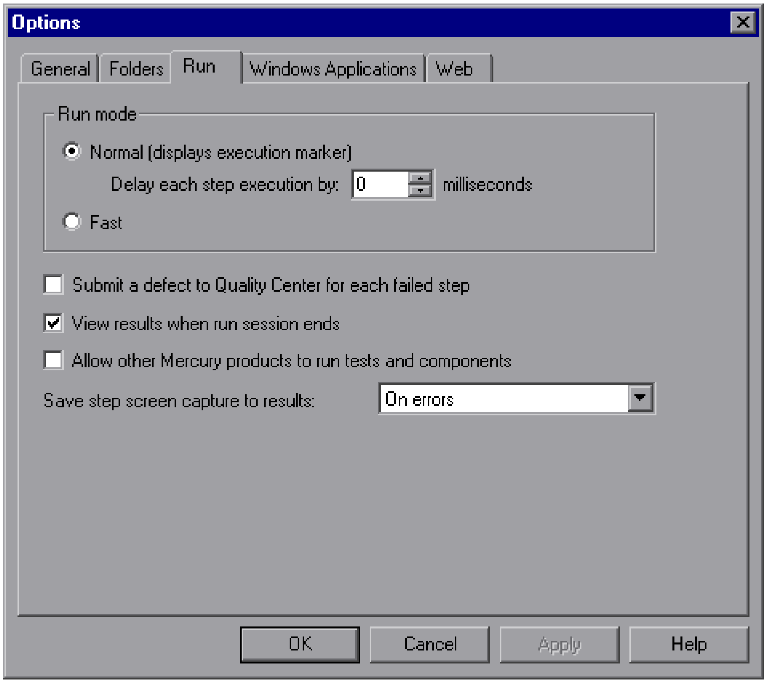
<Current Business Component>

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. **Tip:** 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. When not connected to Quality Center, hold the SHIFT key and click this button. QuickTest adds [QualityCenter], and you enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests. Note that QuickTest searches Quality Center project folders only when you are connected to the corresponding Quality Center project.
×	Deletes the selected folder from the search list.
↑	Moves the selected folder up in the list.
↓	Moves the selected folder down in the list.

Setting Run Testing Options

The Run tab options affect how QuickTest runs components and displays run session results in the Test Results window.



Run mode

● Normal (displays execution marker)

Delay each step execution by: 0 milliseconds

○ Fast

□ Submit a defect to Quality Center for each failed step

☑ View results when run session ends

□ Allow other Mercury products to run tests and components

Save step screen capture to results: On errors

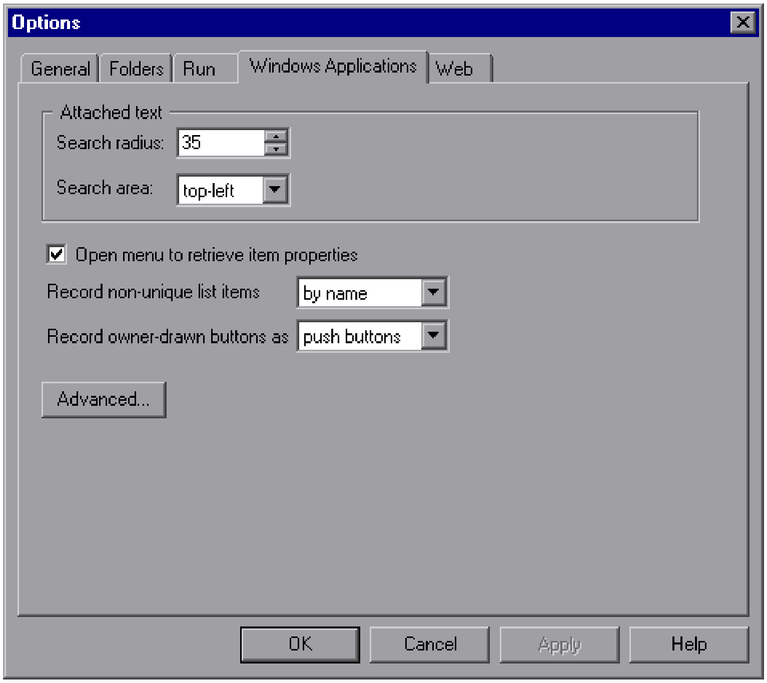
The Run tab includes the following options:

Option	Description
Run mode	Instructs QuickTest how to run your component: • **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. **Delay each step execution by.** You can specify the time in milliseconds that QuickTest should wait before running each consecutive step (up to a maximum of 10000 ms.) The **Normal** run mode option requires more system resources than the **Fast** option, described below. **Note:** You must have Microsoft Script Debugger installed to enable this mode. For more information, refer to the *QuickTest Professional Installation Guide*. • **Fast.** Runs your component without the execution arrow to the left of the Keyword View. This option requires fewer system resources. **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.

Option	Description
Allow other Mercury products to run tests and components	Enables other Mercury products such as Quality Center to run QuickTest components. **Note:** This option is not required to enable WinRunner to run QuickTest components.
Save step screen capture to test results	Relevant only for tests.

Setting Windows Application Testing Options

The Windows Applications tab options enable you to configure how QuickTest records and runs components for Standard Windows, ActiveX, .NET Windows Forms, and Visual Basic applications.



Search radius: 35

Search area: top-left

☑ Open menu to retrieve item properties

Record non-unique list items: by name

Record owner-drawn buttons as: push buttons

The Windows Applications tab includes the following options:

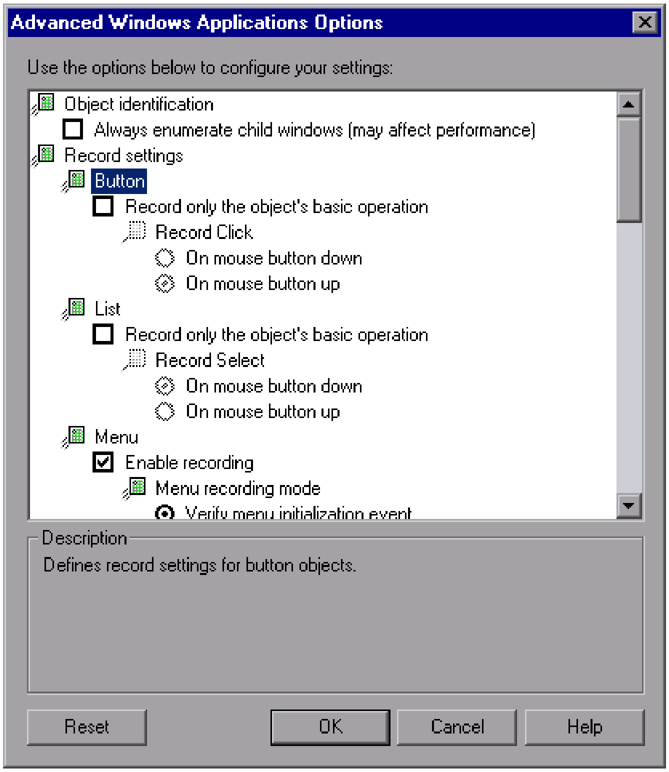
Option	Description
Attached text	Enables you to specify the search criteria that QuickTest uses to retrieve an object's attached text. An object's attached text is the closest static text within a specified radius from a specified point. The retrieved attached text is saved in the object's corresponding text or attached text test object property. **Note:** Sometimes the static text that you believe to be closest to an object is not actually the closest static text. You may need to use trial and error to make sure that the attached text is the static text object of your choice. **Search radius.** Indicates the maximum distance, in pixels, that QuickTest searches for attached text. **Search area.** Indicates the point on an object from which QuickTest searches for the object's attached text. Choose an option from the **Search area** list: • **Top-Left.** Top-left corner • **Top.** Midpoint between the two top corners • **Top-Right.** Top-right corner • **Right.** Midpoint between the two right corners • **Bottom-Right.** Bottom-right corner • **Bottom.** Midpoint between the two bottom corners • **Bottom-Left.** Bottom-left corner • **Left.** Midpoint between the two left corners

Option	Description
Open menu to retrieve item properties	Instructs QuickTest to open menu objects before retrieving menu item properties during a run session. **Note:** Selecting this option may slow the run, but it can be useful if menu item properties change upon opening the menu. This option, selected by default, sets the default behavior for all menu objects. You can use the **ExpandMenu** property in a function library to set this behavior for a specified menu object. For more information, refer to the *QuickTest Professional Object Model Reference*.
Record non-unique list items	Determines what QuickTest records when more than one item (in a list or tree) has an identical name. • **by name.** Records the item's name. When the component runs, QuickTest finds and selects the first instance of the name, regardless of the item chosen during recording. Select this option if the all items with the same name have identical properties. • **by index.** Records the item's index number. Select this option if items with the same name do not necessarily have identical properties.
Record owner-drawn buttons as	Instructs QuickTest how to identify and record custom-made buttons in the application. Choose an option from the list: • **push buttons** • **check boxes** • **radio buttons** • **objects** **Note:** Choosing **objects** records each owner-drawn button as a WinObject.

Option	Description
Advanced	Opens the Advanced Windows Applications Options dialog box, in which you can customize record and run options for your Windows applications. For more information, see "Advanced Windows Applications Options" on page 225.

Advanced Windows Applications Options

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



Advanced Windows Applications Options

Use the options below to configure your settings:

- Object identification
  - □ Always enumerate child windows (may affect performance)
- Record settings
  - Button
    - □ Record only the object's basic operation
    - Record Click
      - ○ On mouse button down
      - ● On mouse button up
  - List
    - □ Record only the object's basic operation
    - Record Select
      - ● On mouse button down
      - ○ On mouse button up
  - Menu
    - ☑ Enable recording
    - Menu recording mode
      - ● Verify menu initialization event

Description

Defines record settings for button objects.

Object Identification Options

You can specify the method QuickTest uses to identify objects when running a component.

The Advanced Windows Applications Options dialog box includes the following **Object identification** options:

Option	Description
Always enumerate child windows (may affect performance)	Instructs QuickTest to enumerate all child windows when recording and running a component. This option is cleared by default and should be used only when an object cannot otherwise be identified, because it may significantly affect performance. For more information, see "Advanced Information" on page 232.

Record Settings Options

You can specify how QuickTest treats certain objects when recording a component.

The Advanced Windows Applications Options dialog box includes the following **Record settings** options:

Category	**Option**
Button	Defines record settings for button objects: • **Record only the object's basic operation.** Enables simplified recording on the button. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 232. • **Record Click.** Specifies whether the **Click** operation should be recorded when the mouse button is pressed (**On mouse button down**) or when the mouse button is released (**On mouse button up**). This option is only enabled when **Record only the object's basic operation** is selected. **Default** = **On mouse button up**.
List	Defines record settings for Windows-based list objects (for example, WinList, WinListView, and VbList): • **Record only the object's basic operation.** Enables simplified recording on the list. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 232. • **Record Select.** Specifies whether the **Select** operation should be recorded when the mouse button is pressed (**On mouse button down**) or when the mouse button is released (**On mouse button up**). This option is only enabled when **Record only the object's basic operation** is selected. **Default** = **On mouse button up**.

Category	**Option**
Menu	Defines record settings for menu objects: - **Enable recording.** Specifies whether QuickTest records operations on menu controls. For example, you may want QuickTest to ignore the actual process of selecting a menu to open another window. This option is selected by default. - **Menu recording mode.** Specifies whether QuickTest verifies or ignores menu initialization events before recording operations on menu controls. This option is only enabled when **Enable recording** is selected. **Default** = **Verify menu initialization event**. For more information, see “Advanced Information” on page 232.
Object	Defines record settings for objects recognized as WinObject test objects: - **Record only the object’s basic operation.** Enables simplified recording on the WinObject test object. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 232. - **Record Click.** Specifies whether the **Click** operation should be recorded when the mouse button is pressed (**On mouse button down**) or when the mouse button is released (**On mouse button up**). This option is only enabled when **Record only the object’s basic operation** is selected. **Default** = **On mouse button down**.

Category	Option
Tab	Defines record settings for tab objects: • **Record only the object's basic operation.** Enables simplified recording on the tab. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 232. • **Record Select.** Specifies whether the **Select** operation should be recorded when the mouse button is pressed (**On mouse button down**) or when the mouse button is released (**On mouse button up**). This option is only enabled when **Record only the object's basic operation** is selected. **Default** = **On mouse button up**.
Toolbar	Defines record settings for toolbar objects: • **Record only the object's basic operation.** Enables simplified recording on the toolbar. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 232. • **Record Press.** Specifies whether the **Press** operation should be recorded when the mouse button is pressed (**On mouse button down**) or when the mouse button is released (**On mouse button up**). This option is only enabled when **Record only the object's basic operation** is selected. **Default** = **On mouse button up**.

Category	Option
Tree view	Defines record settings for tree view objects: • **Record only the object's basic operation.** Enables simplified recording on the tree view. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 232. • **Record Select.** Specifies whether the **Select** operation should be recorded when the mouse button is pressed (**On mouse button down**) or when the mouse button is released (**On mouse button up**). This option is only enabled when **Record only the object's basic operation** is selected. **Default** = **On mouse button up**. • **Record tree items.** Specifies whether tree items are recorded **By name** or **By virtual index**. **Default** = **By name**.
Window	Defines record settings for window objects: • **Record only the object's basic operation.** Enables simplified recording on the window. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 232. • **Record Click.** Specifies whether the **Click** operation should be recorded when the mouse button is pressed (**On mouse button down**) or when the mouse button is released (**On mouse button up**). This option is only enabled when **Record only the object's basic operation** is selected. **Default** = **On mouse button up**.

Category	Option
Keyboard	Defines record settings for operations performed on the keyboard: - **Keyboard state detection.** Specifies which API QuickTest should use to detect the keyboard state. **Default** = **Standard**. For more information, see “Advanced Information” on page 232.
Utility object	Defines record settings for utility objects: - **Record SystemUtil.Run commands.** Specifies whether QuickTest records **SystemUtil.Run** commands when you open an application during a recording session. This option is selected by default. For more information on the **SystemUtil.Run** method, refer to the *QuickTest Professional Object Model Reference*.

Run Settings Options

You can specify how QuickTest treats certain objects when running a component.

The Advanced Windows Applications Options dialog box includes the following **Run settings** options:

Option	Description
Edit Box	Defines run settings for Edit objects: • **Click Edit box before inserting text.** Specifies whether QuickTest performs a **Click** operation to set the focus in an edit box before inserting text in it while running a component. This option is cleared by default. • **Use keyboard events to perform Set operations.** When selected, instructs QuickTest to simulate keyboard events when performing **Set** operations on edit boxes during a run session. When cleared, instructs QuickTest to use API or Window messages for edit box **Set** operations. This option is cleared by default.

Advanced Information

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

Always enumerate child windows

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

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

Record only the object's basic operation

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

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

Keyboard state detection

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

- **Standard.** Uses the GetKeyboardState API to detect the keyboard state. For more information, refer to http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/userinput/keyboardinput/keyboardinputreference/keyboardinputfunctions/getkeyboardstate.asp.
- **Alternate synchronous.** Uses the GetKeyState API to detect the keyboard state. For more information, refer to http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/userinput/keyboardinput/keyboardinputreference/keyboardinputfunctions/getkeystate.asp.
- **Alternate asynchronous.** Uses the GetAsyncKeyState API to detect the keyboard state. For more information, refer to http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/userinput/keyboardinput/keyboardinputreference/keyboardinputfunctions/getasynckeystate.asp.

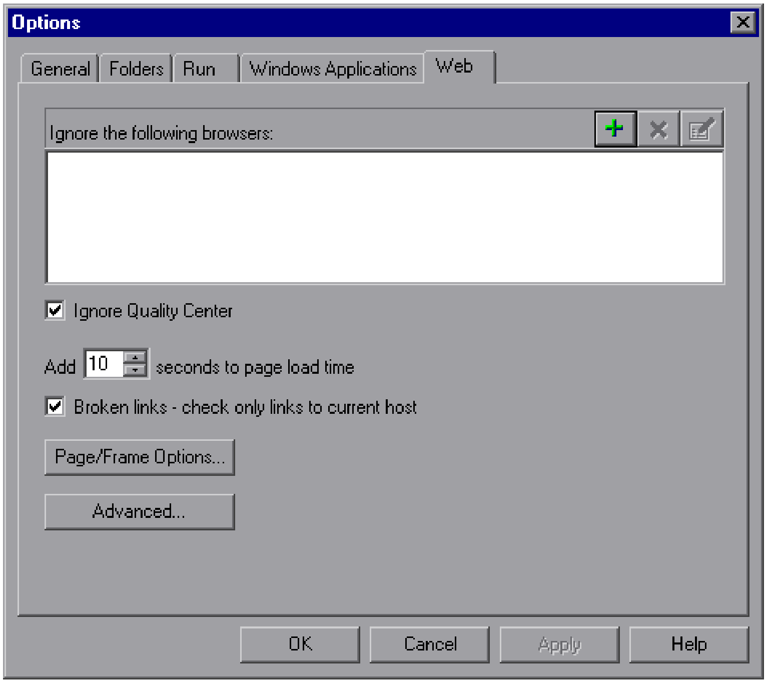
Menu recording mode

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

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

Setting Web Testing Options

The Web tab options determine how QuickTest behaves when recording and running components on Web sites.



Ignore the following browsers:

☑ Ignore Quality Center

Add 10 seconds to page load time

☑ Broken links - check only links to current host

The Web tab includes the following options:

Option	Description
Ignore the following browsers	Instructs QuickTest to ignore any specified browsers that may be open while QuickTest is recording or running a component. For more information, see "Managing the List of Browsers to Ignore" on page 237.
Ignore Quality Center	Instructs QuickTest to ignore all instances of Quality Center that are opened while recording or running a component. By default, this option is selected.
Add __ seconds to page load time	Instructs QuickTest to add a specified number of seconds to the page load time property specified in each Page checkpoint. **Note:** This option is a safeguard that prevents page checkpoints from failing in the event that the amount of time it takes for a page to load during the run is longer than the amount of time it took during the record session.
Broken links - check only links to current host	Instructs QuickTest to check only for broken links that are targeted to your current host.
Page/Frame Options	Opens the Page and Frame Options dialog box, in which you can customize how QuickTest records Page and Frame test objects. For more information, see "Page and Frame Options" on page 241.
Advanced	Opens the Advanced Web Options dialog box, in which you can customize record and run options for your Web sites. For more information, see "Advanced Web Options" on page 244.

Managing the List of Browsers to Ignore

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

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

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

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

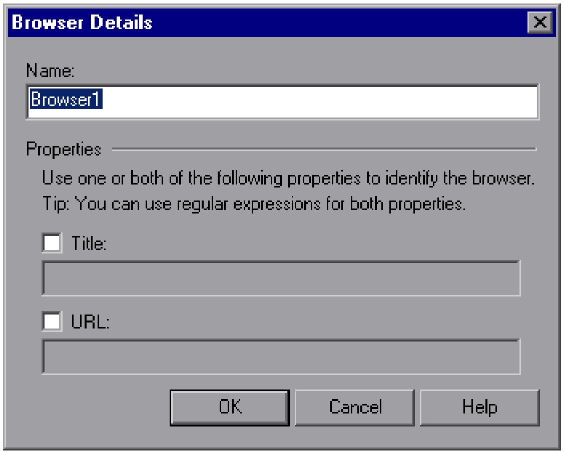
Tip: By default, QuickTest ignores all instances of Quality Center that were opened during a record or run session, if the **Ignore Quality Center** check box in the Web tab of the Options dialog box is selected. There is no need to specify Quality Center in the list of browsers to ignore.

Adding a Browser to the List

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

To add a browser to the list:

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



Browser Details

Name:
Browser1

Properties

Use one or both of the following properties to identify the browser.
Tip: You can use regular expressions for both properties.

□ Title:

□ URL:

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

3 Select one or both of the following properties to identify the browser to be ignored, and then enter the following details:

- **Title**. The name of the Web page as it appears in the title bar of the browser, for example, Yahoo! Finance.*
- **URL**. The URL of the Web page, for example, http://www.finance.yahoo.com Any descendants of this Web page are automatically included in the list of browsers to ignore.

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

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

4 Click **OK**. The browser is added to the list of ignored browsers.

5 Repeat steps 1 to 4 for each browser to be added to the list.

Modifying a Browser in the List

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

To modify a browser in the list:

1 Highlight the browser you want to modify.

2 Click **Edit Browser Details** button. The Browser Details dialog box opens.

3 Make the required changes in the Browser Details dialog box and click **OK**.

Removing a Browser from the List

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

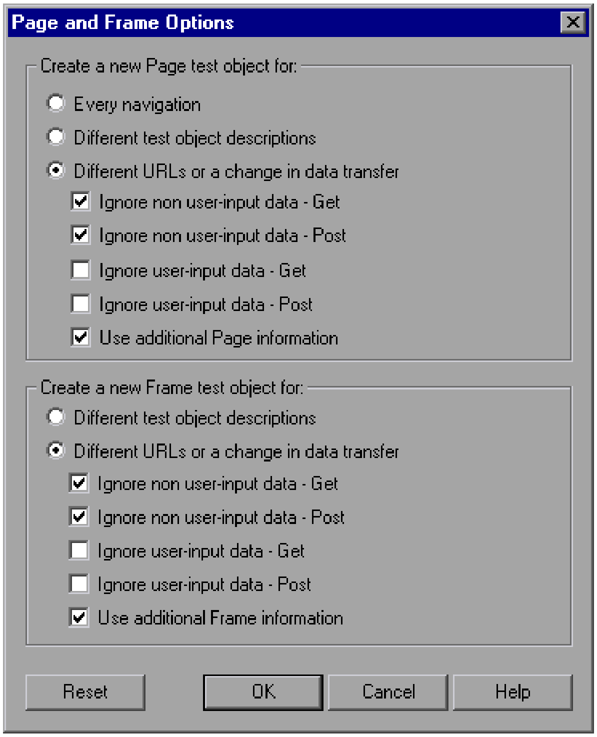
Tip: If a browser in the list is required for running a specific test, you can temporarily remove it from the list by clearing the check mark next to its name in the list of browsers.

To remove a browser from the list:

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

Page and Frame Options

The Page and Frame Options dialog box enables you to modify how QuickTest records Page and Frame objects.



Page and Frame Options

Create a new Page test object for:

○ Every navigation
○ Different test object descriptions
● Different URLs or a change in data transfer
- ☑ Ignore non user-input data - Get
- ☑ Ignore non user-input data - Post
- □ Ignore user-input data - Get
- □ Ignore user-input data - Post
- ☑ Use additional Page information

Create a new Frame test object for:

○ Different test object descriptions
● Different URLs or a change in data transfer
- ☑ Ignore non user-input data - Get
- ☑ Ignore non user-input data - Post
- □ Ignore user-input data - Get
- □ Ignore user-input data - Post
- ☑ Use additional Frame information

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

Page Options

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

Note: These options only affect how Page test objects are created; Frame test objects are created according to the Frame options you select. For more information, see "Frame Options" on page 244.

The following Page options are available:

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

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

- **Different URLs or a change in data transfer.** Creates a new Page object only when the page URL changes, or if the URL stays the same and data that is transferred to the server changes, according to the data types and transfer methods you select (below).

Note: Clear this option to instruct QuickTest to create a new Page test object for every navigation. (QuickTest version 5.6 and earlier worked this way automatically.)

  - **Ignore non user-input data - Get.** Instructs QuickTest to ignore non user-input data if the **Get** method is used to transfer data to the server.

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

  - **Ignore non user-input data - Post.** Instructs QuickTest to ignore non user-input data if the **Post** method is used to transfer data to the server.

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

  - **Ignore user-input data - Get.** Instructs QuickTest to ignore user-input data if the **Get** method is used to transfer data to the server.

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

  - **Ignore user-input data - Post.** Instructs QuickTest to ignore user-input data if the **Post** method is used to transfer data to the server.

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

- **Use additional Page information.** Instructs QuickTest to use additional properties of the test object to identify an existing Page test object.

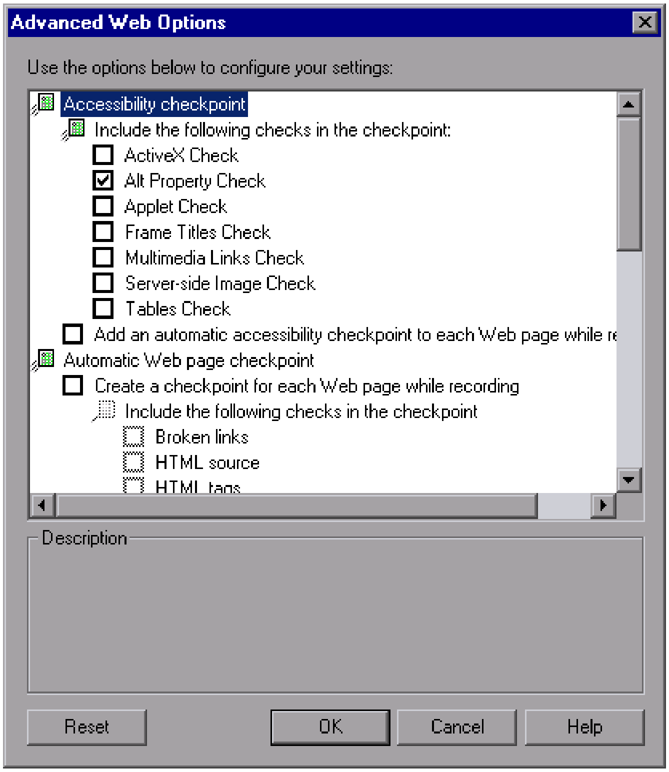
Tip: Select this option to instruct QuickTest to recognize existing pages when the **Back** and **Forward** navigation buttons are used.

Frame Options

The **Create a new Frame test object for** options instruct QuickTest when to create a new Frame object in the object repository while recording. The Frame options are similar to the Page options (except that the **Every navigation** option is not available). For more information, see "Page Options" on page 242.

Advanced Web Options

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



Advanced Web Options

Use the options below to configure your settings:

- Accessibility checkpoint
  - Include the following checks in the checkpoint:
    - □ ActiveX Check
    - ☑ Alt Property Check
    - □ Applet Check
    - □ Frame Titles Check
    - □ Multimedia Links Check
    - □ Server-side Image Check
    - □ Tables Check
  - □ Add an automatic accessibility checkpoint to each Web page while re
- Automatic Web page checkpoint
  - □ Create a checkpoint for each Web page while recording
    - Include the following checks in the checkpoint
      - □ Broken links
      - □ HTML source
      - □ HTML tags

Description

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

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

- **Add an automatic accessibility checkpoint to each Web page while recording**. Instructs QuickTest to automatically add an accessibility checkpoint to each Web page while recording, using the checks selected in the option above.
- **Create a checkpoint for each Web page while recording**. Instructs QuickTest to automatically add a Page checkpoint for each Web page navigated during the recording process.

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

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

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

  Note: If the **Broken links - check only links to current host** option is selected (see “Setting Web Testing Options” on page 235), this number includes only those broken links that are targeted to the current host.

- **HTML source**. Checks that the expected source code is identical to the source code during the run session.
- **HTML tags.** Checks that the expected HTML tags in the source code are identical to those in the run session.
- **Image source.** Checks that the expected source paths of the images are identical to the sources in the run session.
- **Links URL**. Checks that the expected URL addresses for the links are identical to the URL addresses in the source code during the run session.

  - **Load time**. Checks that the expected time it takes for the page to load during the run session is less than or equal to the amount of time it took during the record session PLUS the amount of time specified in the **Add seconds to page load time** option (see "Setting Web Testing Options" on page 235).
  - **Number of images.** Checks that the expected number of images is identical to the number displayed in the run session.
  - **Number of links.** Checks that the expected number of links is identical to the number displayed in the run session.

- **Ignore automatic checkpoints while running tests**. Instructs QuickTest to ignore the automatically added Page checkpoints while running your test.

Record Settings

You can set preferences for recording Web objects. The Advanced Web Options dialog box includes the following **Record** settings:

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

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

- **Record coordinates**. Records the actual coordinates relative to the object for each operation.
- **Record MouseDown and MouseUp as Click.** Records a **Click** method for MouseUp and MouseDown events.
- **Record Navigate for all navigation operations.** Records a **Navigate** statement each time a Frame URL changes.

- **Use standard Windows mouse events.** Instructs QuickTest to use standard Windows mouse events instead of browser events for the following events:
  - **OnClick**
  - **OnMouseDown**
  - **OnMouseUp**

Note: Use this option only if the events are not properly recorded using browser events.

If QuickTest does not record Web events in a way that matches your needs, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as a mouseover that opens a sub-menu, you may need to modify your Web event configuration to recognize such events. For more information, see Chapter 12, "Configuring Web Event Recording."

Run Settings

You can set preferences for working with Web objects during a run session. The Advanced Web Options dialog box includes the following Run settings:

- **Browser cleanup.** Closes all open browsers when the current component closes.

  When this option is selected, all currently open browsers are closed when the current component closes, regardless of whether the browsers were opened before or after QuickTest.
- **Run only click.** Runs Click events using the MouseDown event, the MouseUp event, and the Click event, or using only the Click event.
- **Replay type.** Configures how to run mouse operations according to the selected option:
  - **Event.** Runs mouse operations using browser events.
  - **Mouse.** Runs mouse operations using the mouse.

- **Run using source index.** Uses the source index property for better performance.
- **Resize browser on run if resized during a recording session.** If this option is selected and you resize the browser during a recording session, QuickTest resizes the browser to this size when a subsequent run session begins. At the end of a run session, the browser returns to its default size.

Note: To use this option, select the **Open the following browser** option in the Record and Run Settings dialog box before recording.

When this option is cleared, QuickTest does not change the browser size when a run session begins.

10

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.

Note: For more information on defining component settings in application areas, see Chapter 3, “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 9, "Setting Global Testing Options."

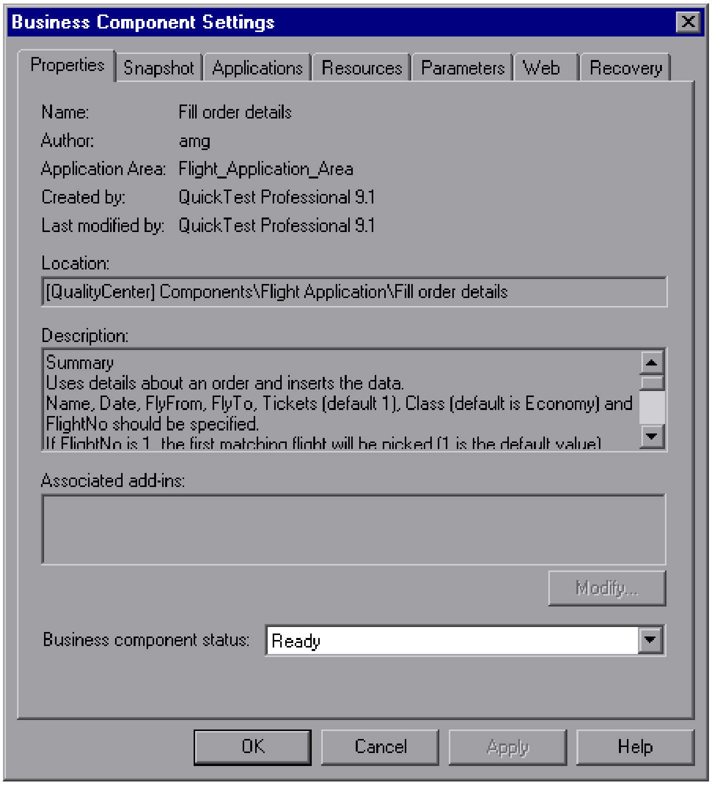
Using the Business Component Settings Dialog Box

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.



Name: Fill order details
Author: amg
Application Area: Flight_Application_Area
Created by: QuickTest Professional 9.1
Last modified by: QuickTest Professional 9.1

Location:
[QualityCenter] Components\Flight Application\Fill order details

Description:
Summary
Uses details about an order and inserts the data.
Name, Date, FlyFrom, FlyTo, Tickets (default 1), Class (default is Economy) and FlightNo should be specified.
If FlightNo is 1, the first matching flight will be picked (1 is the default value)

Associated add-ins:

Business component status: Ready

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.

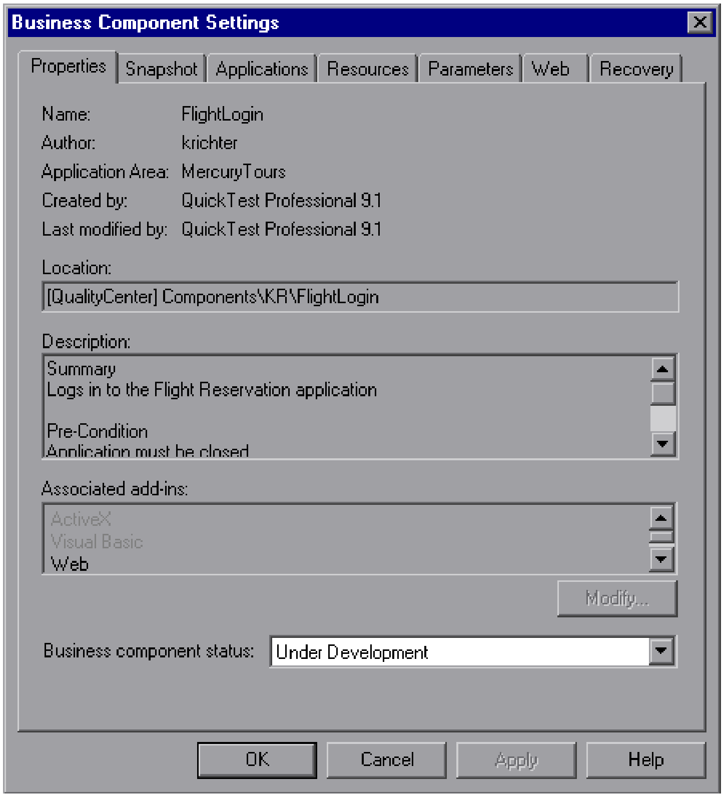
The Business Component Settings dialog box contains the following tabs:

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 255.
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 258.
Applications	The Windows-based applications on which the component can record and run. For more information, see "Viewing Application Settings" on page 260.
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 262.
Parameters	Options for specifying input and output parameters for the component. For more information, see "Defining Parameters for Your Component" on page 263.
Web	Option for specifying the browser navigation timeout for the component. This tab is only displayed when the Web Add-in is installed and loaded. For more information, see "Viewing Web Settings" on page 267.
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 269.

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 *QuickTest Professional User's Guide*. There may also be other tabs corresponding to any external add-ins that are loaded, for example, SAP or Web Services. For information on tabs related to external add-ins, refer to the relevant QuickTest add-in documentation.

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.



Name: FlightLogin
Author: krichter
Application Area: MercuryTours
Created by: QuickTest Professional 9.1
Last modified by: QuickTest Professional 9.1

Location:
[QualityCenter] Components\KR\FlightLogin

Description:
Summary
Logs in to the Flight Reservation application

Pre-Condition
Application must be closed

Associated add-ins:
ActiveX
Visual Basic
Web

Business component status: Under Development

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 116. **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. **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 257.

For information on defining general information for the application area on which your component is based, see "Defining General Settings" on page 77.

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.

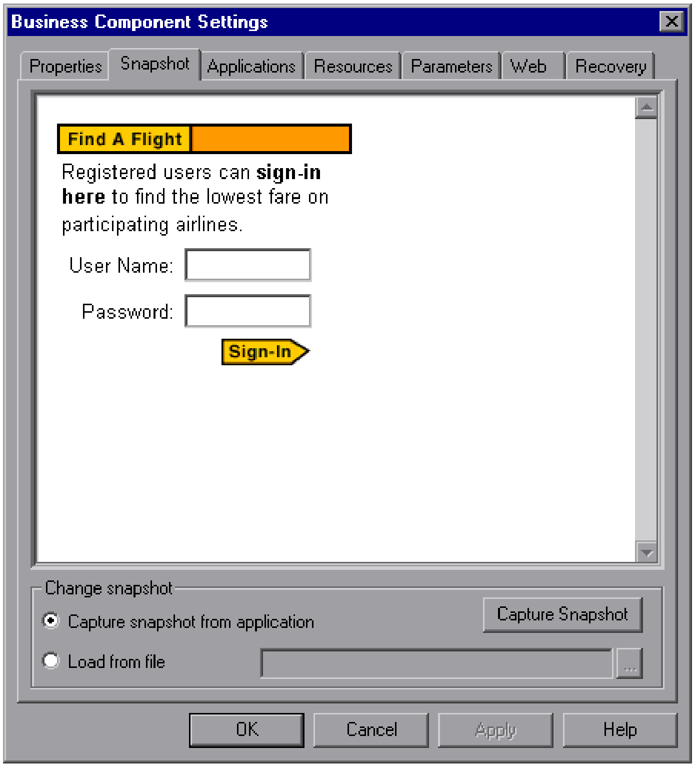
Knowing the status of a business component is important because it affects the status of any business process tests of which it is a part. In general, the component with the most severe status determines the status of the entire business process test. For example, a business component with an **Error** 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.

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 *Mercury Business Process Testing User's Guide*.

The Snapshot tab contains the following options:

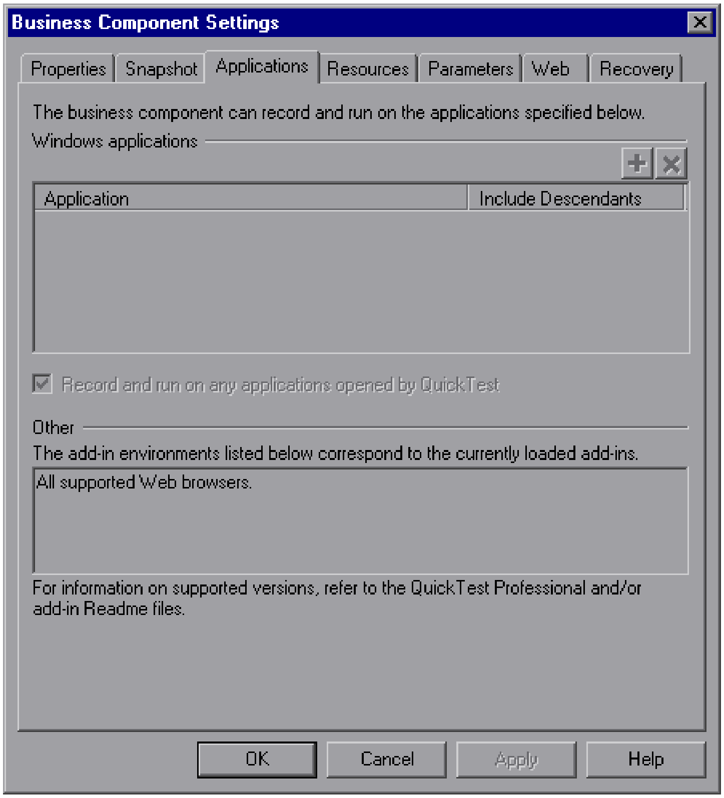
Option	**Description**
Capture snapshot from application	Enables you to define the image to be captured by clicking the **Capture Snapshot** button. You can then drag the crosshairs pointer to select the area to be captured. When you release the mouse button, the captured area is displayed in the Snapshot 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).



The business component can record and run on the applications specified below.

☑ Record and run on any applications opened by QuickTest

The add-in environments listed below correspond to the currently loaded add-ins.

All supported Web browsers.

For information on supported versions, refer to the QuickTest Professional and/or add-in Readme files.

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 99.

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 99.

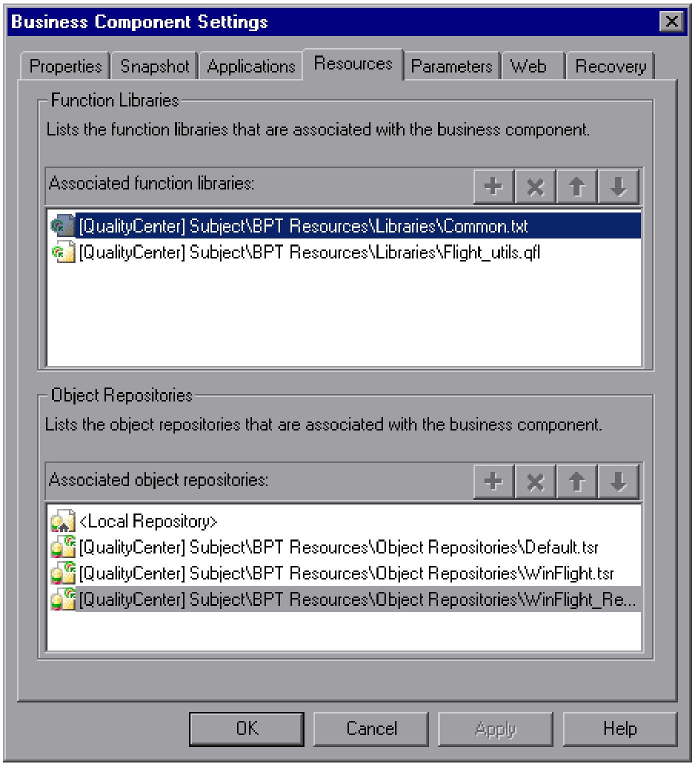
The Applications dialog box and Applications tab may also contain options applicable to any QuickTest external add-ins installed on your computer. For information regarding these options, refer to the documentation provided with the specific add-in.

The Applications tab includes the following items:

Setting	Description
Application	Lists the details of the applications on which to record and run the component. 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.



Function Libraries

Lists the function libraries that are associated with the business component.

Associated function libraries:

[QualityCenter] Subject\BPT Resources\Libraries\Common.txt

[QualityCenter] Subject\BPT Resources\Libraries\Flight_utils.qfl

Object Repositories

Lists the object repositories that are associated with the business component.

Associated object repositories:

<Local Repository>

[QualityCenter] Subject\BPT Resources\Object Repositories\Default.tsr

[QualityCenter] Subject\BPT Resources\Object Repositories\WinFlight.tsr

[QualityCenter] Subject\BPT Resources\Object Repositories\WinFlight_Re...

The Resources tab includes the following items:

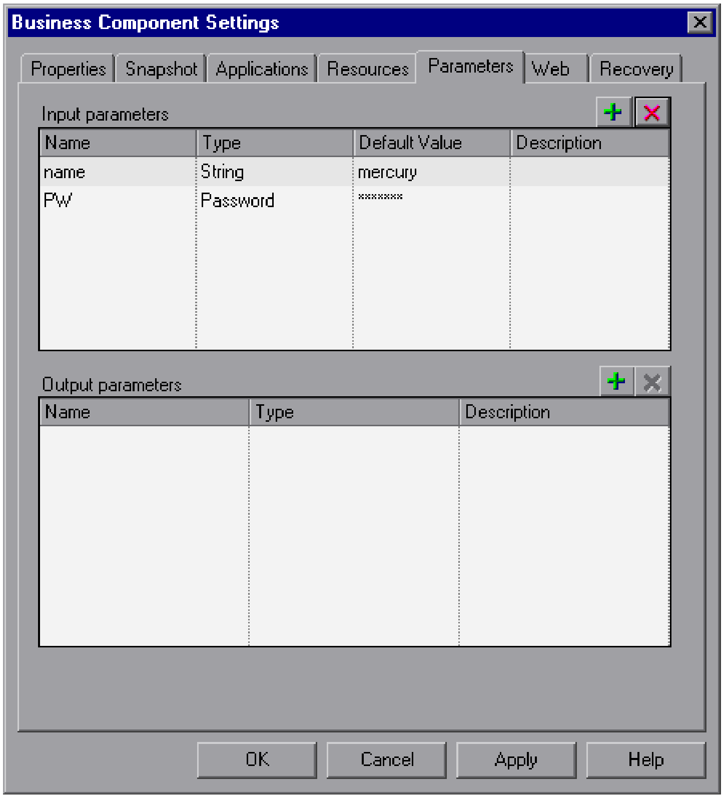
Setting Area	Description
Associated function libraries	Displays the list of function libraries currently associated with your component (via its associated application area). For more information on associating function libraries, see "Managing Function Libraries" on page 83, and "Working with Associated Function Libraries" on page 635.
Object Repositories	Displays the list of shared object repositories currently associated with your component (via its associated application area). 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 88.

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 "Working with Parameters" on page 170. For information on working with component parameters in steps, see "Using Component Parameters in Steps" on page 267.

The Subject Matter Expert can also define component parameters in Quality Center. For information, refer to the *Business Process Testing User's Guide*.



Input parameters

Name	Type	Default Value	Description
name	String	mercury	
PW	Password	*******	

Output parameters

Name	Type	Description

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 *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. 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 531.
Quality Center	Component Iterations dialog box (Test Plan module). For more information, refer to the *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:

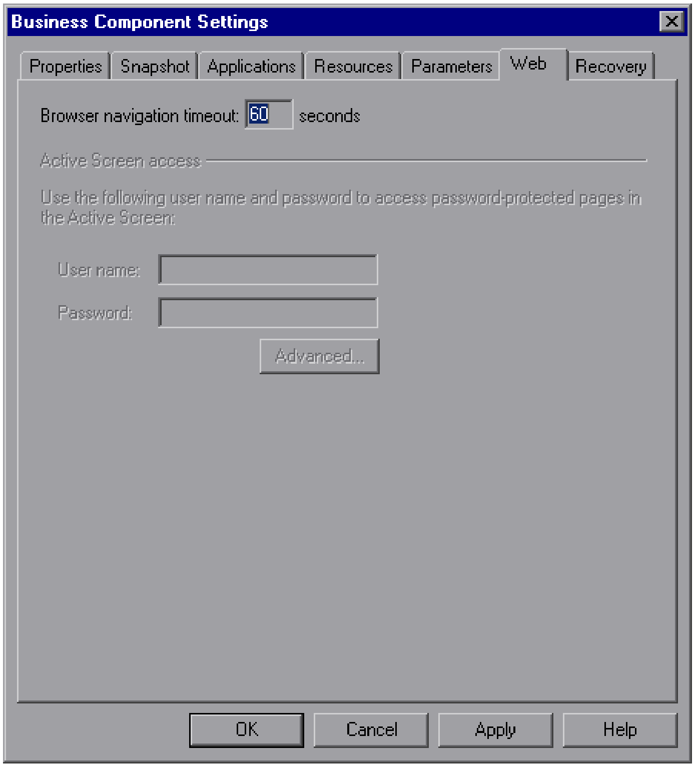
Value Type	QuickTest Default Value
String	Empty string
Boolean	True
Date	The current date
Number	0
Password	Empty string

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 “Working with Parameters” on page 170.

Viewing Web Settings

The Web tab of the Business Component Settings dialog box displays the setting used when running components on an application.



Business Component Settings

Browser navigation timeout: 60 seconds

Active Screen access

Use the following user name and password to access password-protected pages in the Active Screen:

User name:

Password:

You define the Web tab settings for a component in its associated application area settings. For more information, see “Defining Web Settings for Your Application Area” on page 103.

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

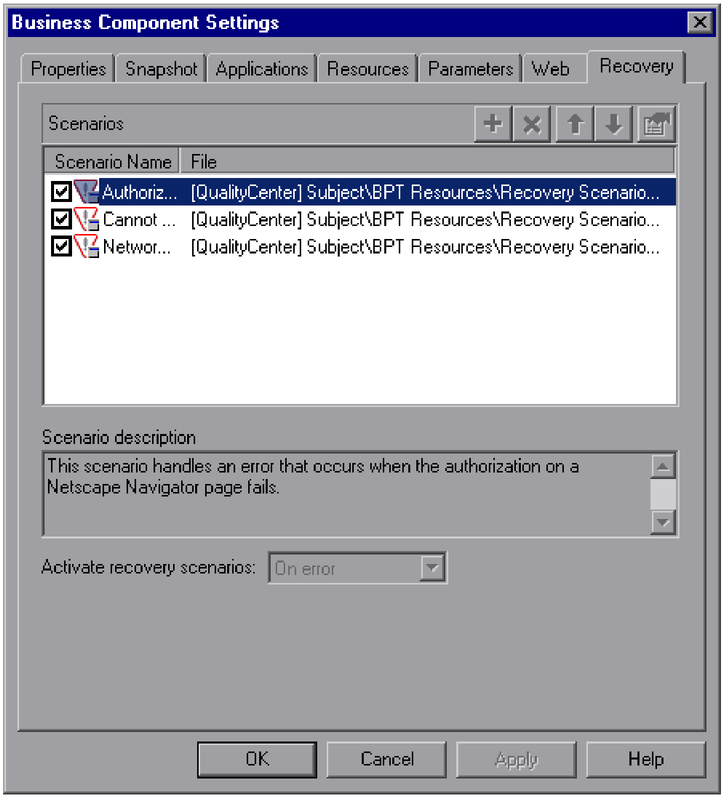
The Web tab includes the following item:

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

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.



	Scenario Name	File
☑	Authoriz...	[QualityCenter] Subject\BPT Resources\Recovery Scenario...
☑	Cannot ...	[QualityCenter] Subject\BPT Resources\Recovery Scenario...
☑	Networ...	[QualityCenter] Subject\BPT Resources\Recovery Scenario...

Scenario description

This scenario handles an error that occurs when the authorization on a Netscape Navigator page fails.

Activate recovery scenarios: On error

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 104.

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 106.
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: • **On every step.** The recovery mechanism is activated after every step. • **On error.** The recovery mechanism is activated only after steps that return an error return value. • **Never.** The recovery mechanism is disabled.

11

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.

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 632.

For more information on working with function libraries, see Chapter 23, "Working with User-Defined Functions and Function Libraries."

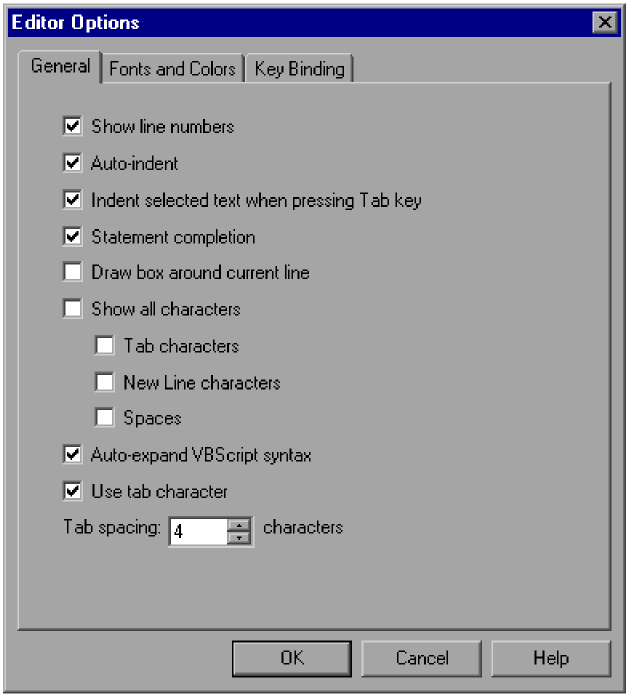
Customizing Editor Behavior

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 23, "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.



☑ Show line numbers
☑ Auto-indent
☑ Indent selected text when pressing Tab key
☑ Statement completion
☐ Draw box around current line
☐ Show all characters
☐ Tab characters
☐ New Line characters
☐ Spaces
☑ Auto-expand VBScript syntax
☑ Use tab character
Tab spacing: 4 characters

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.

Options	Description
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: • **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. • **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. • **a method.** QuickTest displays the syntax for the method, including its specific mandatory and optional arguments. • **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.
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.

Options	Description
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. 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: If Then 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.

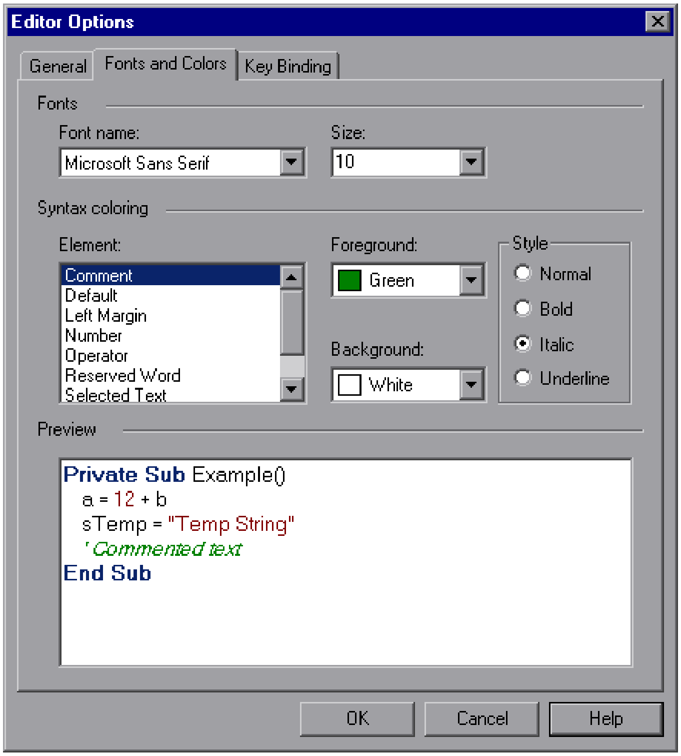
Customizing Element Appearance

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.



Font name: Microsoft Sans Serif

Size: 10

Element:
- Comment
- Default
- Left Margin
- Number
- Operator
- Reserved Word
- Selected Text

Foreground: 🟩 Green

Background: □ White

Style: ○ Normal ○ Bold ● Italic ○ Underline

```
Private Sub Example()
  a = 12 + b
  sTemp = "Temp String"
  'Commented text
End Sub
```

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 steps 4 to 6 for each element you want to modify.
8. Click **OK** to apply the changes and close the dialog box.

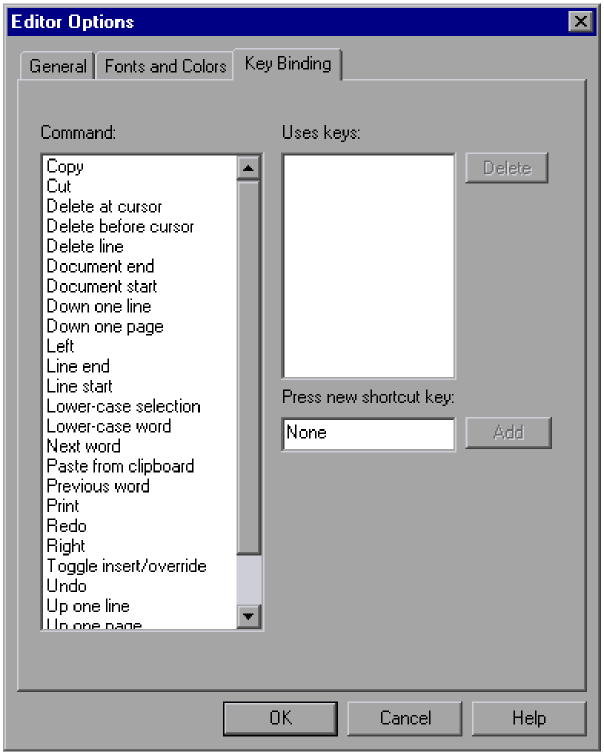
Personalizing Editing Commands

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 Commands Using Shortcut Keys" on page 52.

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.



Command:

Copy
Cut
Delete at cursor
Delete before cursor
Delete line
Document end
Document start
Down one line
Down one page
Left
Line end
Line start
Lower-case selection
Lower-case word
Next word
Paste from clipboard
Previous word
Print
Redo
Right
Toggle insert/override
Undo
Up one line
Up one page

Uses keys:

Press new shortcut key: None

3. Select a command from the **Command** list.
4. Click in the **Press new shortcut key** box and then press the key(s) 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 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 key(s) in the **Uses keys** list, and click **Delete**.

8 Click **OK** to apply the changes and close the dialog box.

12

Configuring Web Event Recording

If QuickTest does not record Web events in a way that matches your needs, you can configure the events you want to record for each type of Web object.

About Configuring Web Event Recording

QuickTest creates your component by recording the events you perform on your Web-based application. An **event** is a notification that occurs in response to an operation, such as a change in state, or as a result of the user clicking the mouse or pressing a key while viewing the document. You may find that you need to record more or fewer events than QuickTest automatically records by default. You can modify the default event recording settings by using the Web Event Recording Configuration dialog box to select one of three standard configurations, or you can customize the individual event recording configuration settings to meet your specific needs.

For example, QuickTest does not generally record mouseover events on link objects. If, however, you have mouseover behavior connected to a link, it may be important for you to record the mouseover event. In this case, you could customize the configuration to record mouseover events on link objects whenever they are connected to a behavior.

Notes:

Event configuration is a global setting and therefore affects all components that are recorded after you change the settings.

Changing the event configuration settings does not affect components that have already been recorded. If you find that QuickTest recorded more or less than you need, change the event recording configuration and then re-record the part of your component that is affected by the change.

Changes to the custom Web event recording configuration settings do not take effect on open browsers. To apply your changes for an existing component, make the changes you need in the Web Event Recording Configuration dialog box, refresh any open browsers, and then start a new recording session.

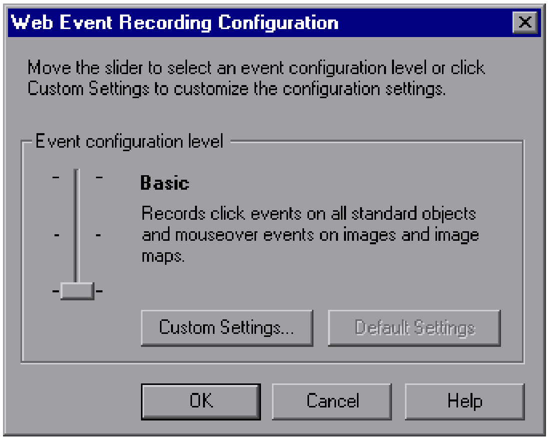
Selecting a Standard Event Recording Configuration

The Web Event Recording Configuration dialog box offers three standard event-configuration levels. By default, QuickTest uses the **Basic** recording-configuration level. If QuickTest does not record all the events you need, you may require a higher event-configuration level.

Level	Description
Basic	**Default** • Always records click events on standard Web objects such as images, buttons, and radio buttons. • Always records the submit event within forms. • Records click events on other objects with a **handler** or **behavior** connected. For more information on handlers and behaviors, see “Listening Criteria” on page 291. • Records the mouseover event on images and image maps only if the event following the mouseover is performed on the same object.
Medium	Records click events on the <DIV>, <SPAN>, and <TD> HTML tag objects, in addition to the objects recorded in the basic level.
High	Records mouseover, mousedown, and double-click events on objects with **handlers** or **behaviors** attached, in addition to the objects recorded in the basic level. For more information on handlers and behaviors, see “Listening Criteria” on page 291.

To set a standard event-recording configuration:

1 Choose **Tools** > **Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.



Move the slider to select an event configuration level or click Custom Settings to customize the configuration settings.

Event configuration level

Basic

Records click events on all standard objects and mouseover events on images and image maps.

2 Use the slider to select your preferred standard event recording configuration.

Tip: You can click the **Custom Settings** button to open the Custom Web Event Recording dialog box where you can customize the event recording configuration. For more information, see “Customizing the Web Event Recording Configuration,” below.

You can click the **Default Settings** button to return the scale to the **Basic** level.

---

3 Click **OK**.

Customizing the Web Event Recording Configuration

If the standard Web event configuration levels do not exactly match your recording needs, you can customize the event recording configuration using the Custom Web Event Recording Configuration dialog box.

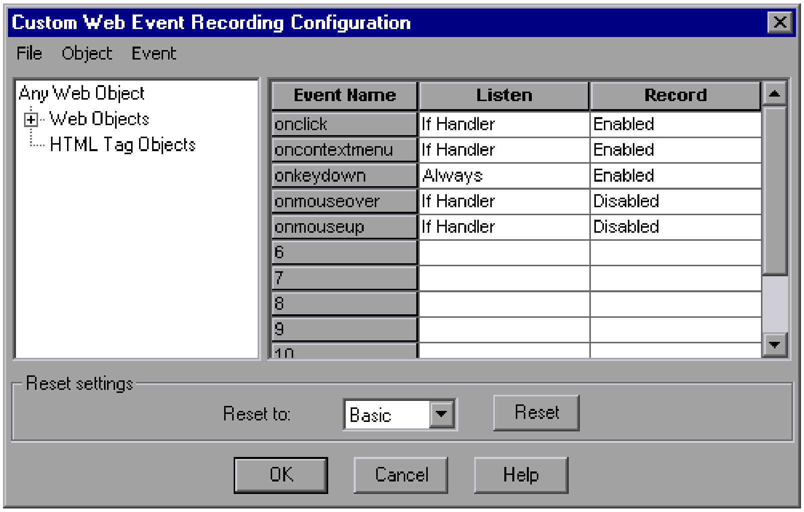
The Custom Web Event Recording Configuration dialog box enables you to customize event recording in several ways. You can:

- Add or delete objects to which QuickTest should apply special listening or recording settings.
- Add or delete events for which QuickTest should listen.
- Modify the listening or recording settings for an event.

To customize the event recording configuration:

1 Choose **Tools** > **Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.

2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.



Event Name	Listen	Record
onclick	If Handler	Enabled
oncontextmenu	If Handler	Enabled
onkeydown	Always	Enabled
onmouseover	If Handler	Disabled
onmouseup	If Handler	Disabled
6		
7		
8		
9		
10		

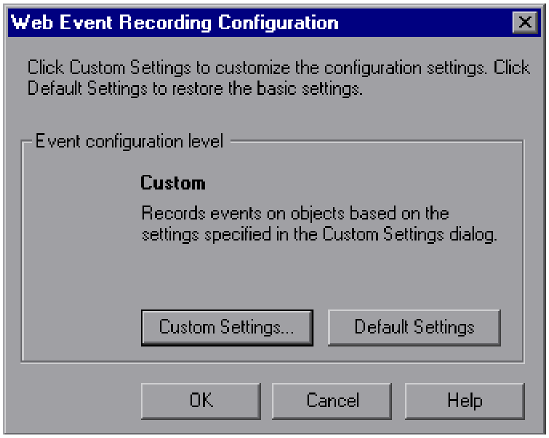
Reset to: Basic

3 Customize the event recording configuration using the following options:

Option	Description
Objects pane	Displays a list of Web test object classes and HTML tag objects. • To add an object, choose **Object** > **Add**. • Only HTML Tag objects can be deleted. To delete an HTML object from the list, choose **Object** > **Delete**. For more information, see "Adding and Deleting Objects in the Custom Configuration Object List" on page 287.
Events pane	Displays a list of events associated with the object. • To add an event to the Events pane, choose **Event** > **Add**. • To delete an event, choose **Event** > **Delete**. For more information, see "Adding and Deleting Listening Events for an Object" on page 289.
Event Name	Specifies the name of the event that QuickTest listens to and/or records, depending on the settings you specify.
Listen	Specifies the criteria that instructs QuickTest when to listen to the event. • **Always.** Always listens to the event. • **If Handler.** Listens to the event if a handler is attached to it. A **handler** is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs. • **If Behavior.** Listens to the event if a DHTML behavior is attached to it. A DHTML **behavior** encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element's default behavior. • **If Handler or Behavior.** Listens to the event if a handler or behavior is attached to it. • **Never.** Never listens to the event. For more information, see "Modifying the Listening and Recording Settings for an Event" on page 291.

Option	Description
Record	Enables or disables recording of the event for the selected object, or enables recording of the event only if the subsequent event occurs on the same object.
Reset	Enables you to reset your settings to a preconfigured level.

4 Click **OK**. The Custom Web Event Recording Configuration dialog box closes. The slider scale on the Web Event Recording Configuration dialog box is hidden and the configuration description displays **Custom**.


Web Event Recording Configuration

Click Custom Settings to customize the configuration settings. Click Default Settings to restore the basic settings.

Event configuration level

Custom

Records events on objects based on the settings specified in the Custom Settings dialog.

5 Click **OK** to close the Web Event Recording Configuration dialog box.

Adding and Deleting Objects in the Custom Configuration Object List

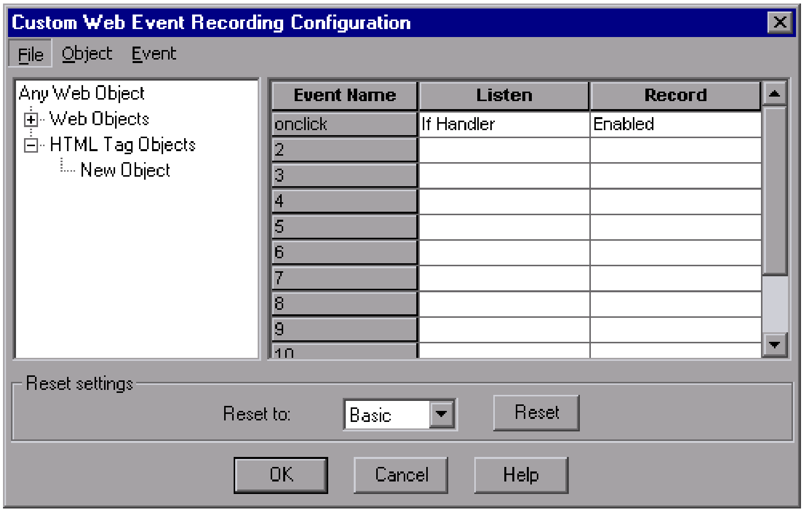
The Custom Web Event Recording Configuration dialog box lists objects in an object hierarchy. The top of the hierarchy is **Any Web Object**. The settings for **Any Web Object** apply to any object on the Web page being tested, for which there is no specific event recording configuration set. Below this are the **Web Objects** and **HTML Tag Objects** categories, each of which contains a list of objects.

When working with the objects in the Custom Web Event Recording Configuration dialog box, keep the following principles in mind:

- If an object is listed in the Custom Web Event Recording Configuration dialog box, then the settings for that object override the settings for **Any Web Object**.
- You cannot delete or add to the list of objects in the **Web Objects** category, but you can modify the settings for any of these objects.
- You can add any HTML Tag object in your Web page to the **HTML Tag Objects** category.

To add objects to the event configuration object list:

1 In the Custom Web Event Recording Configuration dialog box, choose **Object** > **Add**. A **New Object** object is displayed in the HTML Tag Objects list.



Custom Web Event Recording Configuration

Event Name	Listen	Record
onclick	If Handler	Enabled
2		
3		
4		
5		
6		
7		
8		
9		
10		

Reset settings

Reset to: Basic

2 Click **New Object** to rename it. Enter the exact HTML Tag name.

By default the new object is set to listen and record **onclick** events with handlers attached.

For more information on adding or deleting events, see “Adding and Deleting Listening Events for an Object,” below. For more information on listening and recording settings, see “Modifying the Listening and Recording Settings for an Event” on page 291.

To delete objects from the HTML Tag Objects list:

1 From the Custom Web Event Recording Configuration dialog box, select the object in the HTML Tag Objects category that you want to delete.

2 Choose **Object** > **Delete**. The object is deleted from the list.

Note: You cannot delete objects from the **Web Objects** category.

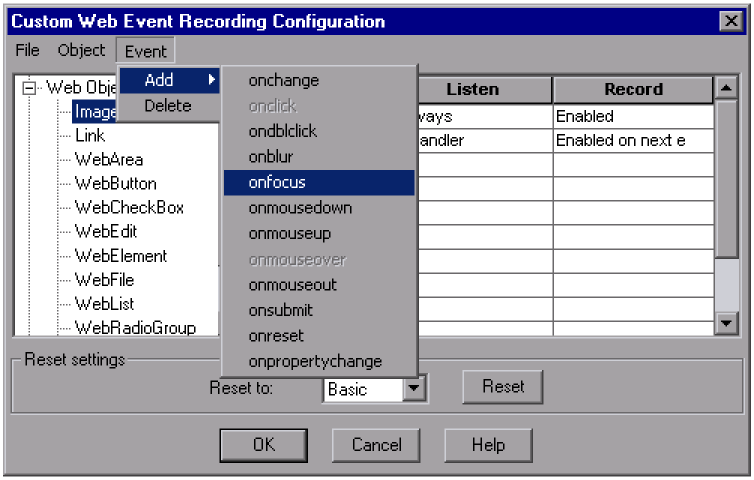
Adding and Deleting Listening Events for an Object

You can add or delete events from the list of events that trigger QuickTest to listen to an object.

To add listening events for an object:

1 In the Custom Web Event Recording Configuration dialog box, select the object to which you want to add an event, or select **Any Web Object**.

2 Choose **Event** > **Add**. A list of available events opens.



3 Select the event you want to add. The event is displayed in the Event Name column in alphabetical order. By default, QuickTest listens to the event when a handler is attached and always records the event (as long as it is listened to at some level).

For more information on listening and recording settings, see “Modifying the Listening and Recording Settings for an Event,” below.

To delete listening events for an object:

1 In the Custom Web Event Recording Configuration dialog box, select the object from which you want delete an event, or select **Any Web Object**.

2 Select the event you want to delete from the **Event Name** column.

3 Choose **Event** > **Delete**. The event is deleted from the **Event Name** column.

Modifying the Listening and Recording Settings for an Event

You can select the listening criteria and set the recording status for each event listed for each object.

Note: The listen and record settings are mutually independent. This means that you can choose to listen to an event for particular object, but not record it, or you can choose not to listen to an event for an object, but still record the event. For more information, see "Tips for Working with Event Listening and Recording" on page 293.

Listening Criteria

For each event, you can instruct QuickTest to listen every time the event occurs on the object if an event handler is attached to the event, if a DHTML behavior is attached to the event, if an event handler or DHTML behavior are attached to the event, or to never listen to the event.

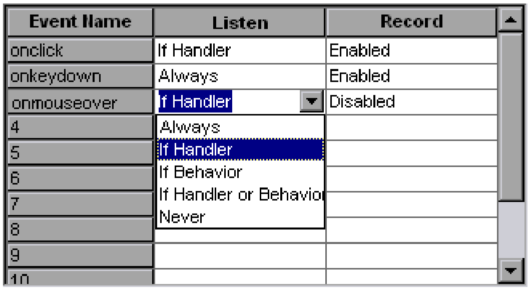
An event **handler** is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs.

A DHTML **behavior** encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element's default behavior.

To specify the listening criterion for an event:

1 From the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the listening criterion or select **Any Web Object**.

2 In the row of the event you want to modify, select the listening criterion you want from the **Listen** column.



Event Name	Listen	Record
onclick	If Handler	Enabled
onkeydown	Always	Enabled
onmouseover	If Handler	Disabled
4		
5		
6		
7		
8		
9		
10		

You can select **Always, If Handler**, **If Behavior**, **If Handler or Behavior**, or **Never**.

Recording Status

For each event, you can enable recording, disable recording, or enable recording only if the next event is dependent on the selected event.

- **Enabled.** Records the event each time it occurs on an object as long as QuickTest listens to the event on the selected object, or on another object to which the event bubbles.

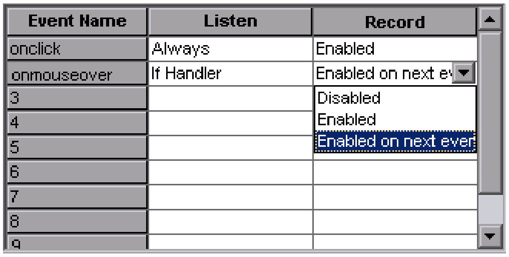
  Bubbling is the process whereby, when an event occurs on a child object, the event can travel up the chain of hierarchy within the HTML code until it encounters an event handler to process the event.

- **Disabled.** Does not record the specified event and ignores event bubbling where applicable.

- **Enabled on next event.** Same as **Enabled**, except that it records the event only if a subsequent event occurs on the same object. For example, suppose a mouseover behavior modifies an image link. You may not want to record the mouseover event each time you happen to move the pointer over this image. It is essential, though, that the mouseover event be recorded before a click event on the same object because only the image that is displayed after the mouseover event enables the link event. This option applies only to the Image and WebArea objects.

To set the recording status for an event:

1. From the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the recording status or select **Any Web Object**.
2. In the row of the event you want to modify, select a recording status from the Record column.



Event Name	Listen	Record
onclick	Always	Enabled
onmouseover	If Handler	Enabled on next ev
3		
4		
5		
6		
7		
8		
9		

Tips for Working with Event Listening and Recording

It can sometimes be difficult to find the ideal listen and recording settings. When defining these settings, keep in mind the following guidelines:

- ➤ If settings for different objects in the Objects Pane conflict, QuickTest gives first priority to settings for specific **HTML Tag Objects** and second priority to **Web Objects** settings. QuickTest only applies the settings for **Any Web Object** to Web objects that were not defined in the **HTML Tag Object** or **Web Objects** areas.
- ➤ To record an event on an object, you must instruct QuickTest to listen for the event, and to record the event when it occurs. You can listen for an event on a child object, even if a parent object contains the handler or behavior, or you can listen for an event on a parent object, even if the child object contains the handler or behavior.

  However, you must enable recording for the event on the **source object** (the object on which the event actually occurs, regardless of which parent object contains the handler or behavior).

For example, suppose a table cell with an **onmouseover** event handler contains two images. When the mouse moves over either of the images, the event also bubbles up to the cell, and the bubbling includes information on the image that the mouse moved over. You can record this mouseover event by:

  - Setting **Listen** on the <TD> tag mouseover event to **If Handler** (so that QuickTest "hears" the event when it occurs), while disabling recording on it, and then setting **Listen** on the <IMG> tag mouseover event to **Never**, while setting **Record** on the <IMG> tag to **Enable** (to record the mouseover event on the image after it is listened to at the <TD> level).
  - Setting **Listen** on the <IMG> tag mouseover event to **Always** (to listen for the mouseover event even though the image tag does not contain a behavior or handler), and setting **Record** on the <IMG> tag to **Enabled** (to record the mouseover event on the image).

- Instructing QuickTest to listen for many events on many objects may lower performance, so it is recommended to limit **Listen** settings to the required objects.
- In rare situations, listening to the object on which the event occurs (the source object) can interfere with the event.

  If you find that your application works properly until you begin recording on the application using QuickTest, your **Listen** settings may be interfering.

  If this problem occurs with a mouse event, try selecting the appropriate **Use standard Windows mouse events** option(s) in the Advanced Web Options dialog box. For more information, see "Advanced Web Options" on page 244.

  If this problem occurs with a keyboard or internal event, or the **Use standard Windows mouse events** option does not solve your problem, set the **Listen** settings for the event to **Never** on the source object (but keep the record setting enabled on the source object), and set the **Listen** settings to **Always** for a parent object.

Recording Right Mouse Button Clicks

QuickTest enables you to record clicks made using left, center, and right mouse buttons. By default, only left clicks are recorded, but you can modify the configuration to record clicks from the right and center buttons, as well.

QuickTest records the **Click** statement when the **OnClick** event is triggered. QuickTest differentiates between the mouse buttons by listening for events configured for each of the mouse buttons. By default, it listens for the **OnMouseUp** event, but you can also configure it to listen for the **OnMouseDown** event using the Web Event Recording Configuration dialog box.

Notes:

Recording of simultaneous clicking of more than one mouse button is not supported.

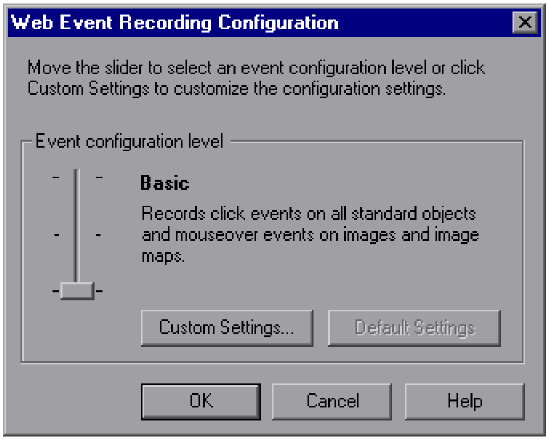
QuickTest does not record the right click that opens the browser context menu, or the selection of an item from the context menu. For more information on modifying the script manually to enable these options, refer to the Mercury Support Knowledge Base (http://support.mercury.com/cgi-bin/portal/CSO/kbBrowse.jsp), select **QuickTest Professional**, and search for Article Number 31270 and Article Number 27184.

Configuring QuickTest to Record Right Mouse Clicks

You instruct QuickTest to record right mouse clicks by modifying the configuration file manually and then loading it.

To configure QuickTest to record right mouse clicks:

1 Choose **Tools** > **Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.



Web Event Recording Configuration

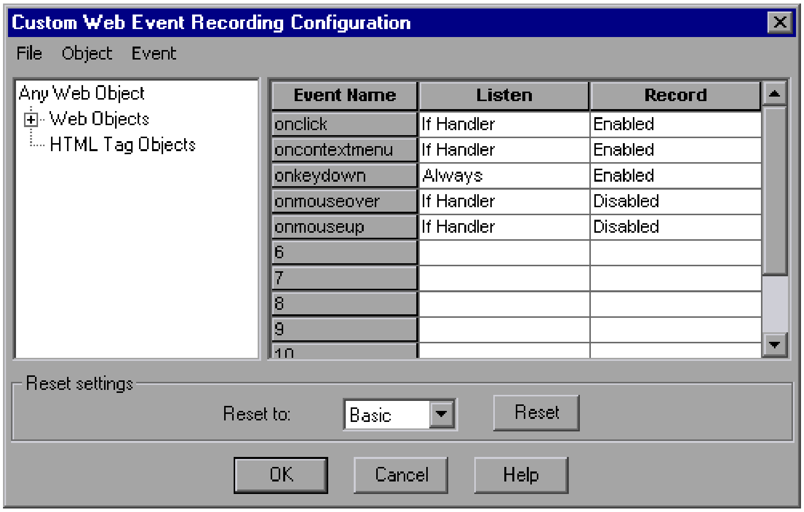
Move the slider to select an event configuration level or click Custom Settings to customize the configuration settings.

Event configuration level

Basic

Records click events on all standard objects and mouseover events on images and image maps.

2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.



Event Name	Listen	Record
onclick	If Handler	Enabled
oncontextmenu	If Handler	Enabled
onkeydown	Always	Enabled
onmouseover	If Handler	Disabled
onmouseup	If Handler	Disabled
6		
7		
8		
9		
10		

Reset to: Basic

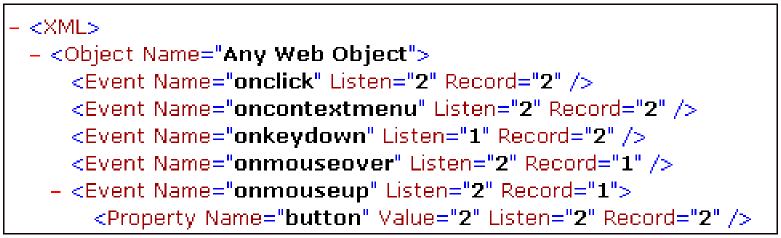
3 In the Custom Web Event Recording Configuration dialog box, choose **File** > **Save Configuration As**. The Save As dialog box opens.

4 Navigate to the folder in which you want to save the web event recording configuration file, and enter a configuration file name. The extension for configuration files is **.xml**.

5 Click **Save** to save the file and close the dialog box.

6 Open the saved configuration file for editing in any text editor. The configuration file uses a defined structure. For more information on the XML file structure, see "Understanding the Web Event Recording Configuration XML Structure" on page 301.

The following example illustrates the beginning of an exported configuration file:



```
- <XML>
  - <Object Name="Any Web Object">
      <Event Name="onclick" Listen="2" Record="2" />
      <Event Name="oncontextmenu" Listen="2" Record="2" />
      <Event Name="onkeydown" Listen="1" Record="2" />
      <Event Name="onmouseover" Listen="2" Record="1" />
    - <Event Name="onmouseup" Listen="2" Record="1">
        <Property Name="button" Value="2" Listen="2" Record="2" />
```

The **Property Name** element controls the recording of the mouse buttons. The values of the mouse buttons are defined as follows:

- **1.** Left
- **2.** Right
- **4.** Middle

7 Edit the file as follows:

- To record a left mouse click for the **onmouseup** event, add the following line:

  <Property Name=”button” Value=”1” Listen=”2” Record=”2”/>

- To record right and left mouse clicks for the **onmousedown** event, add the following lines:

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

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

8 Save the file.

9 In the Custom Web Event Recording Configuration dialog box, choose **File** > **Load Configuration**. The Open dialog box opens.

10 Navigate to the folder in which you saved the edited configuration file, select the file, and click **Open**. The Custom Web Recording Configuration dialog box reopens.

11 Click **OK**. The new configuration is loaded, with all preferences corresponding to those you defined in the XML configuration file. Any Web objects you now record will be recorded according to these new settings.

Saving and Loading Custom Event Configuration Files

You can save the changes you make in the Custom Web Event Recording Configuration dialog box, and load them at any time.

You can also modify the XML file before loading it. For more information on the XML file structure, see “Understanding the Web Event Recording Configuration XML Structure” on page 301.

To save a custom configuration:

1. Customize the event recording configuration as desired. For more information on how to customize the configuration, see “Customizing the Web Event Recording Configuration” on page 285.
2. In the Custom Web Event Recording Configuration dialog box, Choose **File** > **Save Configuration As**. The Save As dialog box opens.
3. Navigate to the folder in which you want to save your event configuration file and enter a configuration file name. The extension for configuration files is **.xml**.
4. Click **Save** to save the file and close the dialog box.

To load a custom configuration:

1. Choose **Tools** > **Web Event Recording Configuration** and then click **Custom Settings** to open the Custom Web Event Recording Configuration dialog box.
2. Choose **File** > **Load Configuration**. The Open dialog box opens.
3. Locate the event configuration file (**.xml**) that you want to load and click **Open**. The dialog box closes and the selected configuration is loaded.

Understanding the Web Event Recording Configuration XML Structure

The Web event recording configuration XML file is structured in a certain format. If you are modifying the file, or creating your own file, you must ensure that you adhere to this format for your settings to take effect.

Following is a sample XML file:

```
<XML>
    <Object Name="Any Web Object">
        <Event Name="onclick" Listen="2" Record="2"/>
        <Event Name="onmouseup" Listen="2" Record="1">
            <Property Name="button" Value="2" Listen="2" Record="2"/>
        </Event>
    </Object>
        . . .
        . . .
        . . .
    <Object Name="WebList">
        <Event Name="onblur" Listen="1" Record="2"/>
        <Event Name="onchange" Listen="1" Record="2"/>
        <Event Name="onfocus" Listen="1" Record="2"/>
    </Object>
</XML>
```

You define the listening criteria and recording status options in the XML using the following possible values:

Settings	Possible Values
Listen	**1.** Always **2.** If Handler **4.** If Behavior **6.** If Handler or Behavior **0.** Never
Record	**1.** Disabled **2.** Enabled **6.** Enabled on Next Event

Resetting Event Recording Configuration Settings

You can restore standard settings after you set custom settings by resetting the event recording configuration settings to the basic level from the Web Event Recording Configuration dialog box. You can also restore the default custom level settings from the Custom Web Event Recording Configuration dialog box.

Note: When you choose to reset standard settings, your custom settings are cleared completely. If you do not want to lose your changes, be sure to save your settings in an event configuration file. For more information, see “Saving and Loading Custom Event Configuration Files” on page 300.

To reset basic level configuration settings from the Web Event Recording Configuration dialog box:

1 Choose **Tools** > **Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.

2 Click **Default**. The standard configuration slider is displayed again and all event settings are restored to the **Basic** event recording configuration level.

3 If you want to select a different standard configuration level, see “Selecting a Standard Event Recording Configuration” on page 283.

You can also restore the settings to a specific (base) custom configuration from within the Custom Web Event Recording Configuration dialog box so that you can begin customizing from that point.

To reset the settings to a custom level from the Custom Web Event Recording Configuration dialog box:

1 Choose **Tools** > **Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.

2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.

3 In the **Reset to** box, select the standard event recording level you want.

4 Click **Reset**. All event settings are restored to the defaults for the level you selected.

Part IV

Working with Test Objects and Object Repositories

13

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, so that you can easily add statements to your script in the Expert View or use test objects and methods in your functions, when creating operations for components.

About Understanding the Test Object Model

QuickTest tests your dynamically changing application by learning and identifying test objects and their expected properties and values. During recording, QuickTest analyzes each object in your application 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 While Recording

QuickTest learns objects just as you would. For example, suppose as part of an experiment, Johnny 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. Johnny 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, Johnny 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 Johnny and points out one of three children sitting on a picnic blanket. Johnny 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, Johnny 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 during the recording process.

First, it “looks” at the object on which you are recording and stores it as a **test object**, determining in which test object class it fits. Just as Johnny immediately checked whether the item was a person, animal, plant, or thing. 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 Johnny planned to learn before seeing the picture. When you record on an object, QuickTest 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; like when Johnny 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 Johnny 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, Johnny 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 Johnny 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 while recording. It expects to find a perfect match for both the mandatory and any assistive properties it used to create a unique description while recording. As long as the object in the application does not change significantly, the description learned during recording 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 Johnny's experiment. In this phase, the tester shows Johnny 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. Johnny 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, Johnny 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. Johnny 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 recorded 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 15, "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 record for 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.

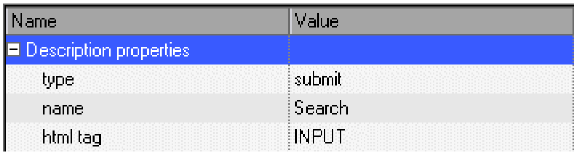
When you perform an operation on your application while recording, QuickTest:

- identifies the QuickTest test object class that represents the object on which you performed the operation 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
- records the operation that you performed on the object using the appropriate QuickTest test object method

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

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

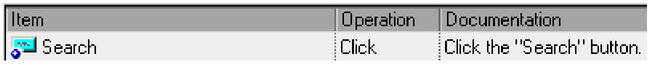
QuickTest identifies the object that you clicked as a **WebButton** test object. It creates a WebButton object with the name Search, and records the following properties and values for the **Search** WebButton:



Name	Value
⊟ Description properties	
type	submit
name	Search
html tag	INPUT

It also 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:



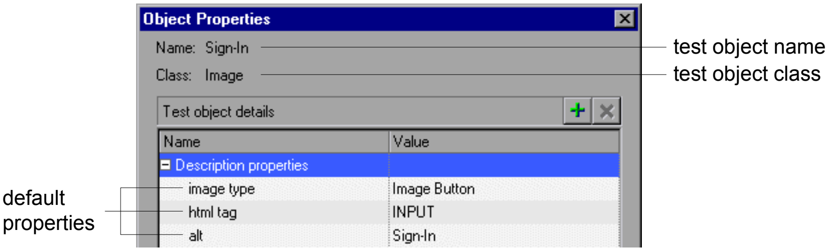
Item	Operation	Documentation
Search	Click	Click the "Search" button.

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 records and it 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 on which you record an operation.



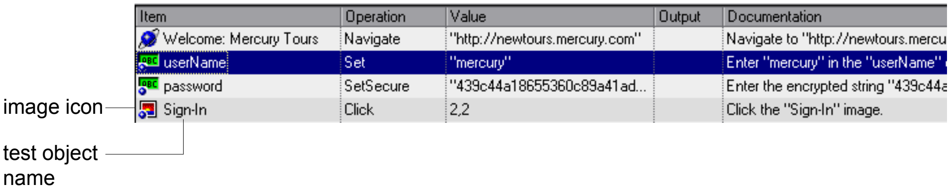
Object Properties

Name: Sign-In

Class: Image

Test object details

Name	Value
⊟ Description properties	
image type	Image Button
html tag	INPUT
alt	Sign-In



Item	Operation	Value	Output	Documentation
Welcome: Mercury Tours	Navigate	"http://newtours.mercury.com"		Navigate to "http://newtours.mercu
userName	Set	"mercury"		Enter "mercury" in the "userName"
password	SetSecure	"439c44a18655360c89a41ad...		Enter the encrypted string "439c44a
Sign-In	Click	2,2		Click the "Sign-In" image.

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 record descriptions of the objects in your application, and you can enable and configure the Smart Identification mechanism. For more information, see Chapter 15, "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 forth).

Similarly, test object methods are methods that QuickTest recognizes and records when they are performed on an object while you are recording, and that QuickTest performs when your component runs. 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 700.

Each test object method you perform while recording is recorded as a separate step in your component. When you run your component, QuickTest performs the recorded test object method on the run-time object.

Test object properties are the properties whose values are captured from the objects in your Web site or application when you record your component. 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. To make the test object property values match the property values of the run-time object, 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 14, "Working with Test 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. You can view the current test object property values of any object on your desktop using the Properties tab of the Object Spy. For information on the Object Properties and Object Repository dialog boxes, see "Modifying Test Object Properties" on page 340. For information on viewing test object property values using the Object Spy, see "Viewing Object Properties Using the Object Spy" on page 316.

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 Methods and Method Syntax Using the Object Spy" on page 319.

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 700.

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 701.

For more information on test object methods and properties refer to the *Mercury QuickTest Professional Object Model Reference*.

Viewing Object Properties Using the Object Spy

Using the Object Spy, you can view the properties of any object in an open application. You use the Object Spy pointer to point to an object. The Object Spy displays the selected object's hierarchy tree and its properties and values in the Properties tab of the Object Spy dialog box.

To view object properties:

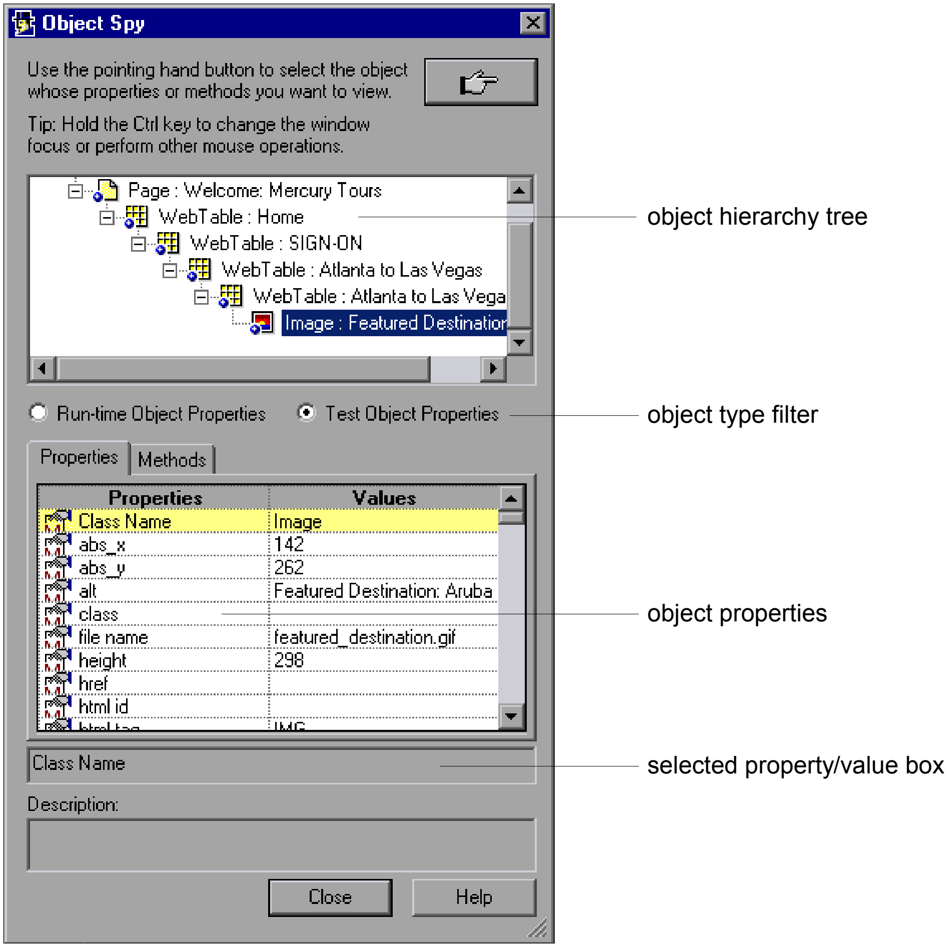
1. Open your browser or application to the page containing the object on which you want to spy.
2. Choose **Tools** > **Object Spy** or click the **Object Spy** toolbar button to open the Object Spy dialog box and display the **Properties** tab. Alternatively, click the **Object Spy** button from the Object Repository dialog box. For more information on the Object Repository dialog box, see "Understanding the Object Repository Window" on page 329.
3. In the Object Spy dialog box, click the pointing hand. Both QuickTest and the Object Spy are minimized so that you can point to and click on any object in the open application.

Note: 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 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 9, "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.

---

4 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.

5 Click the object whose properties you want to view. The Object Spy returns to focus and displays the object hierarchy tree and the properties of the object that is selected within the tree.



Object Spy

Use the pointing hand button to select the object whose properties or methods you want to view.

Tip: Hold the Ctrl key to change the window focus or perform other mouse operations.

```
Page : Welcome: Mercury Tours
  WebTable : Home
    WebTable : SIGN-ON
      WebTable : Atlanta to Las Vegas
        WebTable : Atlanta to Las Vega
          Image : Featured Destinatior
```

○ Run-time Object Properties ● Test Object Properties

Properties | Methods

Properties	Values
Class Name	Image
abs_x	142
abs_y	262
alt	Featured Destination: Aruba
class	
file name	featured_destination.gif
height	298
href	
html id	
html tag	IMG

Class Name

Description:

object hierarchy tree

object type filter

object properties

selected property/value box

Tip: 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.

6 To view the properties of the test object, click the **Test Object Properties** radio button. To view the properties 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. For more information, see "Retrieving Run-Time Object Properties" on page 702.

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 property value of the objects in your application during the run session. For more information, see "Retrieving and Setting Test Object Property Values" on page 700.

7 If you want to view properties for another object within the displayed tree, click the object in the tree.

8 If you want to copy an object property or value to the Clipboard, click the property or value. The value is displayed in the selected property/value box. Highlight the text in the selected property/value box and use CTRL + C to copy the text to the Clipboard or right-click the highlighted text and choose **Copy** from the menu.

Viewing Object Methods and Method Syntax Using the Object Spy

In addition to viewing object properties, the Object Spy also enables you to view both the run-time object methods and the test object methods associated with an object and to view the syntax for a selected method. You use the Object Spy pointer to point to an object. The Object Spy displays the object hierarchy tree and the run-time object methods or test object methods associated with the selected object in the Methods tab of the Object Spy dialog box.

To view object methods:

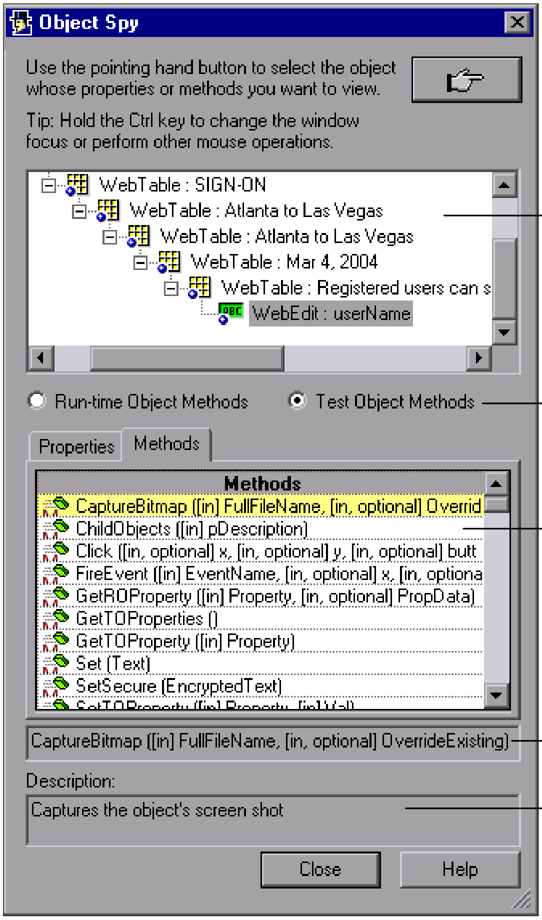
1. Open your browser or application to the page containing the object on which you want to spy.
2. Choose **Tools** > **Object Spy** or click the **Object Spy** toolbar button to open the Object Spy dialog box. Alternatively, click the **Object Spy** button from the Object Repository dialog box. For more information on the Object Repository dialog box, see “Understanding the Object Repository Window” on page 329.
3. Click the **Methods** tab.
4. Click the pointing hand. Both QuickTest and the Object Spy are minimized so that you can point to any object on the open application.

Note: If the object you want 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 this option in the Options dialog box. For more information, see Chapter 9, “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.

---

5 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.

6 Click the object whose associated methods you want to view. The Object Spy returns to focus and displays the object hierarchy tree and the run-time object or test object methods associated with the object that is selected in the tree.



Object Spy

Use the pointing hand button to select the object whose properties or methods you want to view.

Tip: Hold the Ctrl key to change the window focus or perform other mouse operations.

```
WebTable : SIGN-ON
  WebTable : Atlanta to Las Vegas
    WebTable : Atlanta to Las Vegas
      WebTable : Mar 4, 2004
        WebTable : Registered users can s
          WebEdit : userName
```

○ Run-time Object Methods   ● Test Object Methods

Properties | Methods

Methods
CaptureBitmap ([in] FullFileName, [in, optional] Overrid
ChildObjects ([in] pDescription)
Click ([in, optional] x, [in, optional] y, [in, optional] butt
FireEvent ([in] EventName, [in, optional] x, [in, optiona
GetROProperty ([in] Property, [in, optional] PropData)
GetTOProperties ()
GetTOProperty ([in] Property)
Set (Text)
SetSecure (EncryptedText)
SetTOProperty ([in] Property, [in] Val)

CaptureBitmap ([in] FullFileName, [in, optional] OverrideExisting)

Description:

Captures the object's screen shot

object hierarchy tree

object type filter

object methods

selected method syntax box

method description box

Tip: You can resize the Object Spy dialog box. This is useful if you have a deep hierarchy, or long syntax, enabling you view all the information without scrolling.

---

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

Tip: You can use the **Object** property to activate the run-time object methods displayed in the Object Spy. For more information, see "Activating Run-Time Object Methods" on page 702.

---

8 If you want to view methods for another object within the displayed tree, click the object on the tree.

9 If you want to copy the syntax of a method to the Clipboard, click the method in the list. The syntax is displayed in the selected method syntax box. Highlight the text in the selected method syntax box and use CTRL + C to copy the text to the Clipboard, or right-click the highlighted text and choose **Copy** from the menu.

14

Working with Test Objects

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

About Working with Test 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**.

As QuickTest learns the test objects, it stores them in the component's **local object repository**. You can choose to keep the stored objects in the local object repository, or you can choose to store the objects in a **shared object repository**. Storing the objects in the local object repository makes them available only to the specific component, but not to other components. Storing the objects in one or more shared object repositories enables multiple components (via their application areas) to use them. 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 325.

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 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 16, "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 manipulate some aspects of a local object repository using the QuickTest Object Repository automation object model. For example, you can add, remove, and rename test objects in the local object repository. For more information, see "Manipulating Object Repositories Using Automation" on page 448.

Understanding Object Repository Types

Test objects can be stored in two types of object repositories—a shared object repository and a local object repository. A **shared object repository** stores test 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 326.

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. Test 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 88.

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 record and run 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 test object properties.

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

- You are creating components using keyword-driven methodologies (not using record).
- 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 test 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 328.) 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.
- As you record operations on objects in your application, QuickTest automatically stores the information about those objects in the corresponding local object repository (if the 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 an 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 begins adding test objects to the local object repository as you record or learn objects.

- If you learn or record on the same object in your application in two different components, the 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 327.)

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 the objects in your component. Otherwise, the component may fail. For more information, see "Adding Objects to the Object Repository" on page 361.

You modify a shared object repository using the Object Repository Manager. For more information, see Chapter 16, "Managing Object Repositories."

When working with a shared object repository:

- If you record operations on an object that already exists in either the shared or local object repository, QuickTest uses the existing information and does not add the object to the 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.
- QuickTest does not add an object to the shared object repository as you record operations on it. Instead, it adds new objects to the local object repository (not the shared object repository) as you learn objects or record steps on them (unless those same objects already exist in an associated shared object repository).

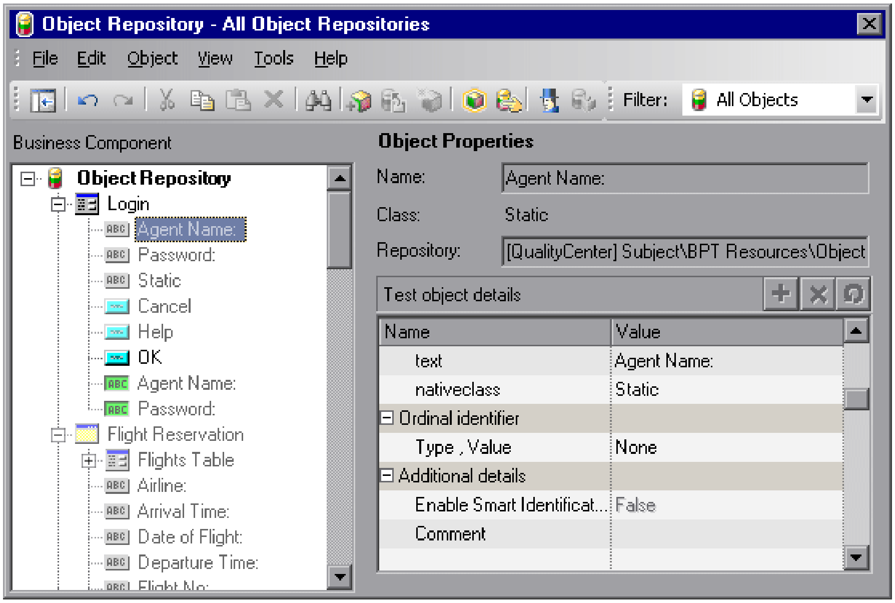
You can export the local objects to a shared object repository. For more information, see “Exporting Local Objects to an Object Repository” on page 382.

You can also merge the local objects directly to a shared object repository that is associated with the same component. This can 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 468.

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 objects in the current component (including all local objects and all objects in any shared object repositories associated with the selected component).



Filter: All Objects

Name: Agent Name:

Class: Static

Repository: [QualityCenter] Subject\BPT Resources\Object

Name	Value
text	Agent Name:
nativeclass	Static
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identificat...	False
Comment	

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

While the Object Repository window is open, you can continue using QuickTest, and you can continue modifying test 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 test object details area. For more information, see "Showing and Hiding the Test Object Details Area" on page 335.

You can use the Object Repository window to view the test object description of any test object in the repository (in local and shared object repositories), to modify local test objects and their properties, and to add objects to your local object repository.

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 or redo operations that were performed before the save operation.

For more information on viewing and modifying test object properties, see "Modifying Test Object Properties" on page 340.

Note: Even when steps containing a test 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 415.

The Object Repository window contains the following information:

Information	Description
Business Component	Indicates that the current testing document is a business component.
Object repository tree	Contains all objects in the current component (all local objects and all objects in any shared object repositories associated with the selected component). **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 88. You can filter the objects shown in the object repository tree. For more information, see "Filtering the Object Repository Window" on page 335.
Name	The name that QuickTest assigns to the test object. You can change the name of a test object in the local object repository. For more information, see "Renaming Test Objects" on page 346.
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.
Test object details	Enables you to view the properties and property values used to identify an object during a run session. You can also modify the test object details for a test object in the local object repository. For more information, see "Understanding the Test Object Details Area" on page 332. You can choose whether to show or hide the test object details area. For more information, see "Showing and Hiding the Test Object Details Area" on page 335.

Understanding the Test Object Details Area

The **Test object details** area 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.

Tip: You can choose whether to show or hide the test object details area. For more information, see "Showing and Hiding the Test Object Details Area" on page 335.

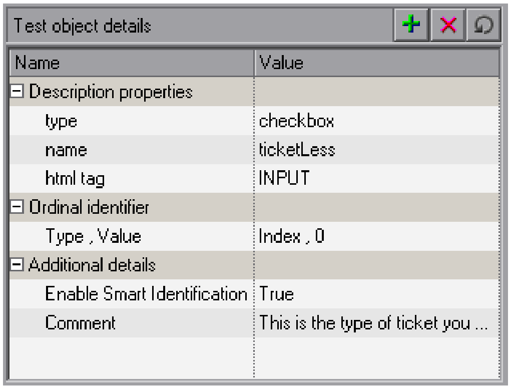
In the Object Repository window, objects in a shared object repository are displayed in the Object Properties pane (including the **Test object details** area) in read-only format. To modify objects in a shared object repository, open the shared object repository using the Object Repository Manager. For more information, see Chapter 16, "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 337.

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 339.

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 Using the Object Spy" on page 316.

You can modify test object 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.



Test object details

Name	Value
⊟ Description properties	
type	checkbox
name	ticketLess
html tag	INPUT
⊟ Ordinal identifier	
Type , Value	Index , 0
⊟ Additional details	
Enable Smart Identification	True
Comment	This is the type of ticket you ...

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 **Test object details** area contains the following items:

Item	Description
Description properties	The properties and property values used to identify the object during a run session. 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 348. 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 342.
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 355.
Additional details	Contains the following options: • **Enable Smart Identification.** Enables you to select **True** or **False** to specify whether QuickTest should use Smart Identification to identify the test object during the run session if it is not able to identify the object using the test object description. **Note:** This option is available only if Smart Identification properties are defined for the test object's class in the Object Identification dialog box. For more information on Smart Identification, see "Configuring Smart Identification" on page 401. • **Comment.** Enables you to add textual information about the test object.

Showing and Hiding the Test 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 test object details area.

To change the Object Repository window view mode:

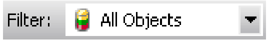
Perform one of the following, depending on the mode you want to show:

- 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.



Filter: All Objects

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 Web sites and applications change, you may need to change the property values of the steps in your component. Suppose an object in your application changes. 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 337
- "Viewing Object Properties and Property Values" on page 339
- "Modifying Test Object Properties" on page 340
- "Specifying or Modifying Property Values" on page 342
- "Updating Test Object Properties from an Object in Your Application" on page 344
- "Restoring Default Properties for a Test Object" on page 346
- "Renaming Test Objects" on page 346
- "Adding Properties to a Test Object Description" on page 348
- "Defining New Test Object Properties" on page 352
- "Removing Properties from a Test Object Description" on page 354
- "Specifying Ordinal Identifiers" on page 355

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:

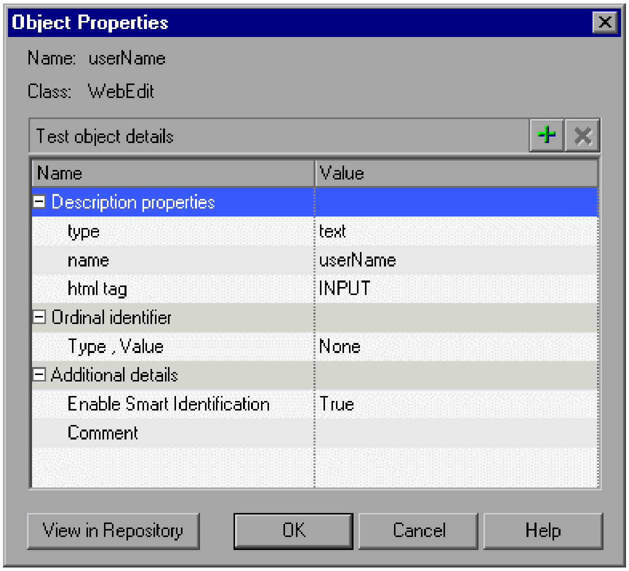
1. In the Object Repository window, select an object from a shared object repository that you want to copy to the local object repository. Objects in a shared object repository are colored gray. You can select more than one object to copy, as long as the selected objects have the same parent objects.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



Object Properties

Name: userName

Class: WebEdit

Test object details

Name	Value
⊟ Description properties	
type	text
name	userName
html tag	INPUT
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	True
Comment	

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 340.
- 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 an object by modifying one or more of the object's property values or by changing the set of properties used to identify that object. You can do this for objects in the local object repository using the Object Repository window, and for objects in the shared object repository using the Object Repository Manager.

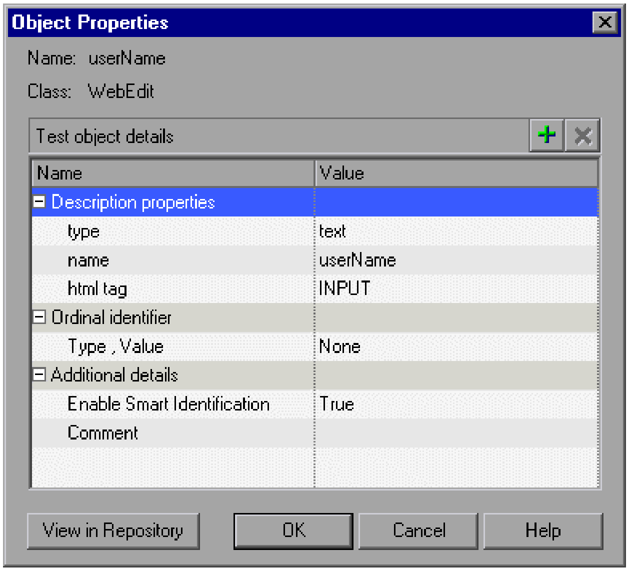
You can also automatically update the description of one or more test objects in your object repository based on the actual updated object properties in your application. For more information, see "Updating Test Object Properties from an Object in Your Application" on page 344.

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 Using the Object Spy" on page 316.

To modify an object property:

1 Right-click the step containing the 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.



Name: userName

Class: WebEdit

Test object details

Name	Value
⊟ Description properties	
type	text
name	userName
html tag	INPUT
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	True
Comment	

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 Test Object Details Area" on page 332 and "Viewing and Modifying Test Object Properties" on page 336.

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

You can also find and replace specific object property values. For more information, see "Finding Objects in an Object Repository" on page 374.

Note: In some cases, the Smart Identification mechanism may enable QuickTest to identify an object, even when some of its property values change. However, if you know about property value changes for a specific object, you should try to correct the object definition so that QuickTest can identify the object from its basic object description. For more information on the Smart Identification mechanism, see Chapter 15, "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 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 Using the Object Spy" on page 316.

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 an 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 A.* icon is displayed next to the value. For information on parameterizing values, see “Working with Parameters” on page 170.
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 an 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 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 368.

You can do this for objects in the local object repository using the Object Repository window, and for 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 minimized and the cursor changes to a pointing hand, so that you can point to and click on any object in the open application.

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 9, "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 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.

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 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 objects in the local object repository using the Object Repository window. You rename 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 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 object in the selected component. If other components in your business process test also include operations on the local object, you should modify the object's name in each relevant component.

When you modify the name of an object in the local object repository, the name is automatically updated for all occurrences of the object. When you modify the name of an 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 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 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 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 object in the selected node in the tree. Then press ENTER or click anywhere else to remove the focus from the object.
- Press F2 and enter the new name for the object.
- In the **Name** box in the Object Properties pane, enter the new name for the object. Then click anywhere else to remove the focus from the object.

Note: The name you assign to the 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 object. You can use the Add Properties dialog box to change the properties that are included in the object description. You can do this for objects in the local object repository using the Object Repository window or Object Properties dialog box, and for 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 352.

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 were not automatically learned while recording. 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 object that was recorded so that you 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 Using the Object Spy" on page 316.

Note: You can also modify the set of properties that QuickTest learns when it records objects from a particular object class using the Object Identification dialog box. Such a change generally affects only objects that you learn or record after you make the change. For more information, see "Configuring Object Identification" on page 385. 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" on page 533.

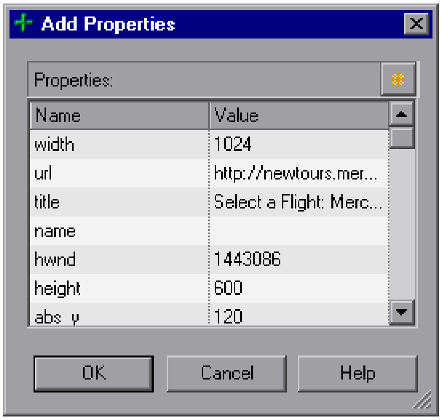
To add properties to a test object description:

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 an 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.



Name	Value
width	1024
url	http://newtours.mer...
title	Select a Flight: Merc...
name	
hwnd	1443086
height	600
abs_y	120

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 352.

---

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 342.

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 objects in the local object repository using the Object Repository window or Add Properties dialog box, and for 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 Using the Object Spy" on page 316.

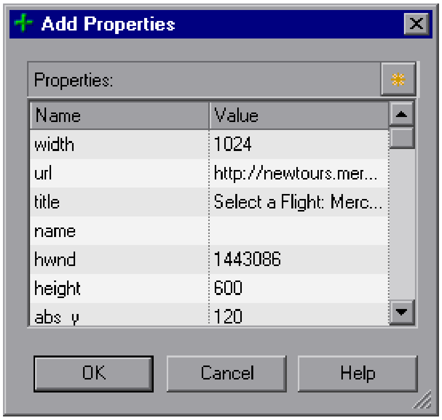
To define a new test object property:

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.

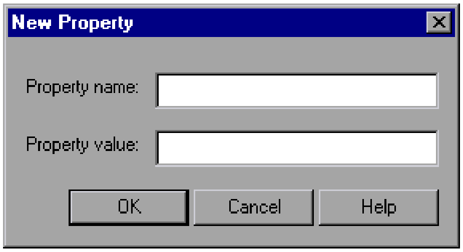
Tip: For an object in the local object repository, you can also select the required test object and choose **Edit** > **Step Properties** > **Object Properties**, click the **Add description properties** button, and then perform the following steps in the Add Properties dialog box.

The Add Properties dialog box opens.



Name	Value
width	1024
url	http://newtours.mer...
title	Select a Flight: Merc...
name	
hwnd	1443086
height	600
abs_y	120

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

For more information on ordinal identifiers, see "Selecting an Ordinal Identifier" on page 394.

To specify an ordinal identifier:

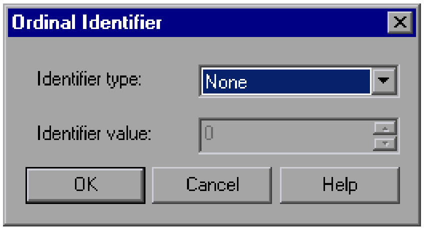
1 Select the test object whose ordinal identifier you want to specify.

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.



Ordinal Identifier

Identifier type: None

Identifier value: 0

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 no ordinal identifier was recorded or learned.

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.

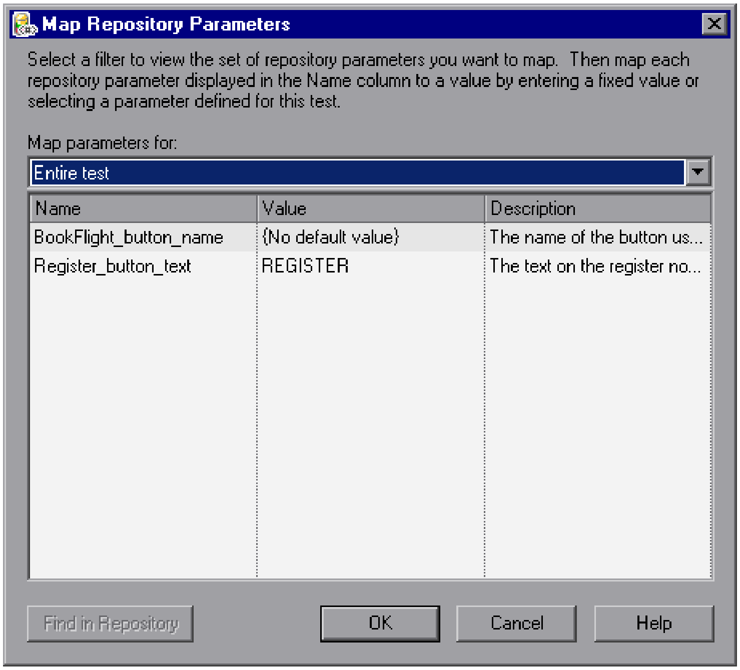
Before you map repository parameters, if you have more than one repository parameter with the same name in different shared object repositories that are associated with the same component, the repository parameter from the shared object repository with the highest priority (as defined in the shared object 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.



Map Repository Parameters

Select a filter to view the set of repository parameters you want to map. Then map each repository parameter displayed in the Name column to a value by entering a fixed value or selecting a parameter defined for this test.

Map parameters for: Entire test

Name	Value	Description
BookFlight_button_name	{No default value}	The name of the button us...
Register_button_text	REGISTER	The text on the register no...

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 7, "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: • **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)
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. You can perform one of the following: • Enter a new constant value. • Parameterize the value by clicking in the **Value** cell of the relevant parameter and then clicking the parameterization button <#>. • 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 434.

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, refer to the *Mercury 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, refer to the *Mercury 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 436.

4 Repeat step 3 for any additional parameter values that you want to map. Then click **OK** to close the Map Repository Parameter dialog box.

Adding Objects to the Object Repository

When you create a shared object repository for your keyword-driven testing infrastructure, you can add objects to it in a number of different ways. You can choose whether to add only a selected object, or to add all objects of a certain type, such as all button objects, or to add all 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 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 an object to the local object repository only if that object does not already exist in a shared object repository that is associated with the component. If an 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 337.

If needed, you can merge test objects from the local object repository into a shared object repository. For more information, see Chapter 17, "Merging Shared Object Repositories."

You can also add objects to a shared object repository while navigating through your application. For more information, see "Adding Objects Using the Navigate and Learn Option" on page 431.

Tip: You can also add an 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 154.

Adding an Object Using the Add Objects to Local or Add Objects Option

You can add objects to your object repository directly from your application. You can choose to add a specific 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 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 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 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 minimized and the cursor becomes 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 9, “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.

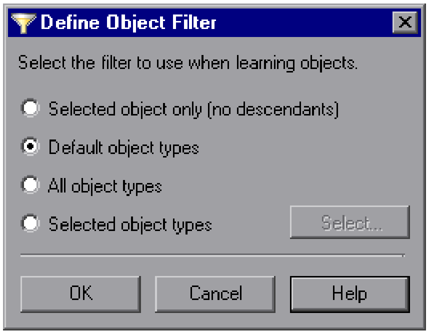
---

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.



Define Object Filter

Select the filter to use when learning objects.

○ Selected object only (no descendants)

● Default object types

○ All object types

○ Selected object types

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 366.

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.

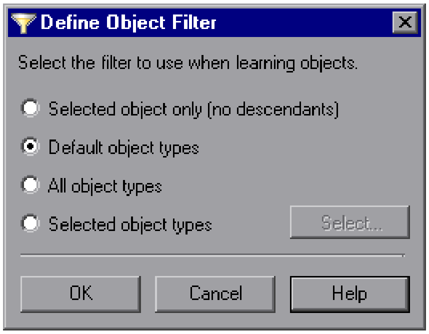
QuickTest also adds the new object's parent objects if they do not already exist in the object repository. Local objects are shown in black in the object repository tree to indicate they are editable; shared objects are shown in gray and can only be edited in the Object Repository Manager.

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 336.

Understanding the Define Object Filter Dialog Box

When adding an 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.



Define Object Filter

Select the filter to use when learning objects.

○ Selected object only (no descendants)

● Default object types

○ All object types

○ Selected object types

You can choose from the following options:

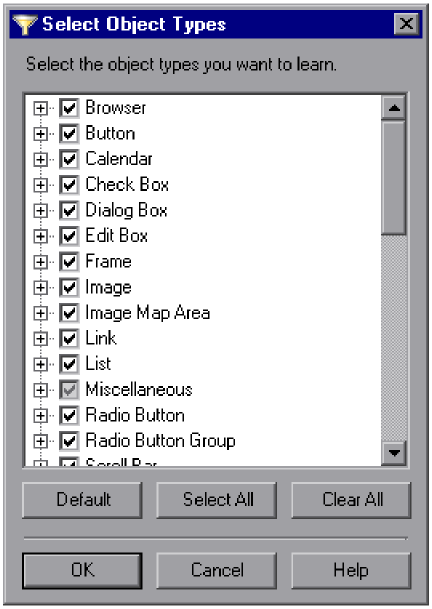
- **Selected object only (no descendants).** Adds to the object repository the previously selected object's properties and values, without its descendant objects.
- **Default object types.** Adds to the object repository the previously selected object's properties and values, with the properties and 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," below.

Understanding the Select Object Types Dialog Box

The Select Object Types dialog box enables you to specify a custom object filter for adding 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 external 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 external add-in automatically filters out certain objects, refer to your add-in documentation.

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 or Web site. 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.

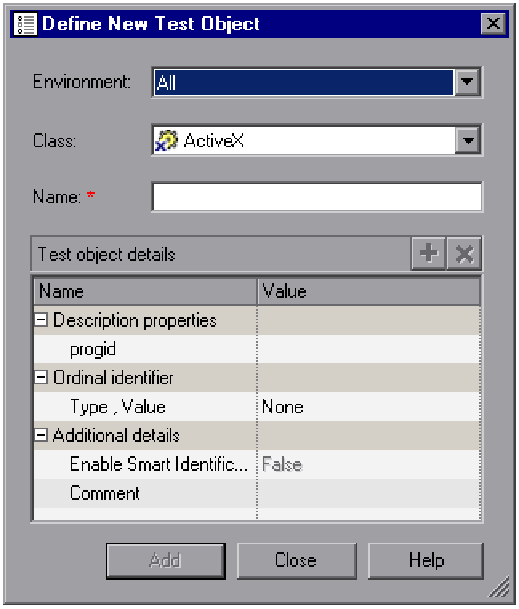
When you define a new object in the object repository as described in this section, the object is added to the local object repository and can only be used by the current component. If you want to add the object to the shared object repository so that it can be used in multiple components, you must add it using the Object Repository Manager. For more information, see Chapter 16, “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 344.

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.



Define New Test Object

Environment: All

Class: ActiveX

Name: *

Test object details

Name	Value
⊟ Description properties	
progid	
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identific...	False
Comment	

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 “Selecting QuickTest Add-Ins” on page 197.

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 336.

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

8 Repeat steps 3 to 7 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 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 337.

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.

You can copy, paste, and move objects in the following ways.

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

Drag it 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 it 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. When you drag an object, by default, 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 a valid recorded hierarchy.
- 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.

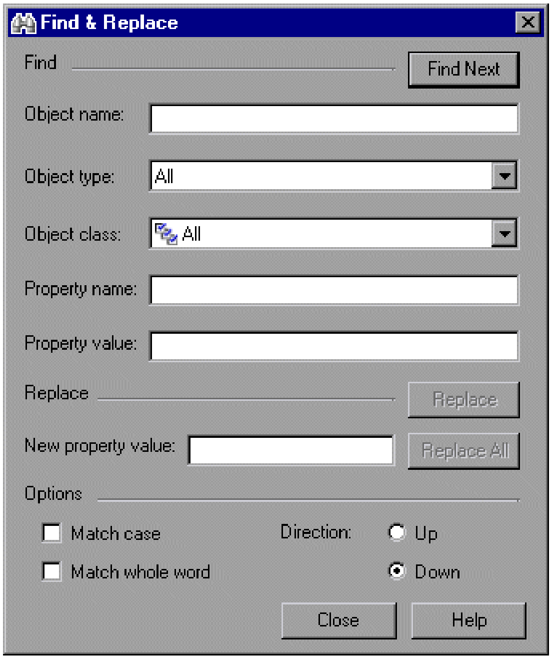
You replace property values for objects in the local object repository using the Object Repository window. You replace property values for objects in shared object repositories using the Object Repository Manager.

Note: 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.



Find & Replace

Find

Object name:

Object type: All

Object class: All

Property name:

Property value:

Replace

New property value:

Options

□ Match case

□ Match whole word

Direction: ○ Up ● Down

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 an object in your object repository and highlight it in the application or Web site you are testing. When you choose to highlight an object, QuickTest indicates the selected object's location in your application by temporarily showing a blue 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 or Web site is open to the correct window or page.
2. Select the 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 blue border in the application or Web site.

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

Locating an Object in the Object Repository

You can select an object in the application or Web site 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.

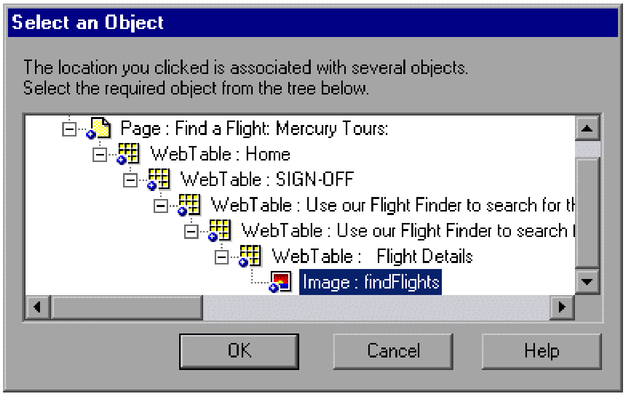
To locate an object in the object repository:

1. Make sure your application or Web site is open to the correct window or page.
2. Click the **Locate in Repository** button or choose **View** > **Locate in Repository**. QuickTest is minimized, and your cursor changes to a pointing hand.

3 Use the pointing hand to click on the required object in your application or Web site.

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.

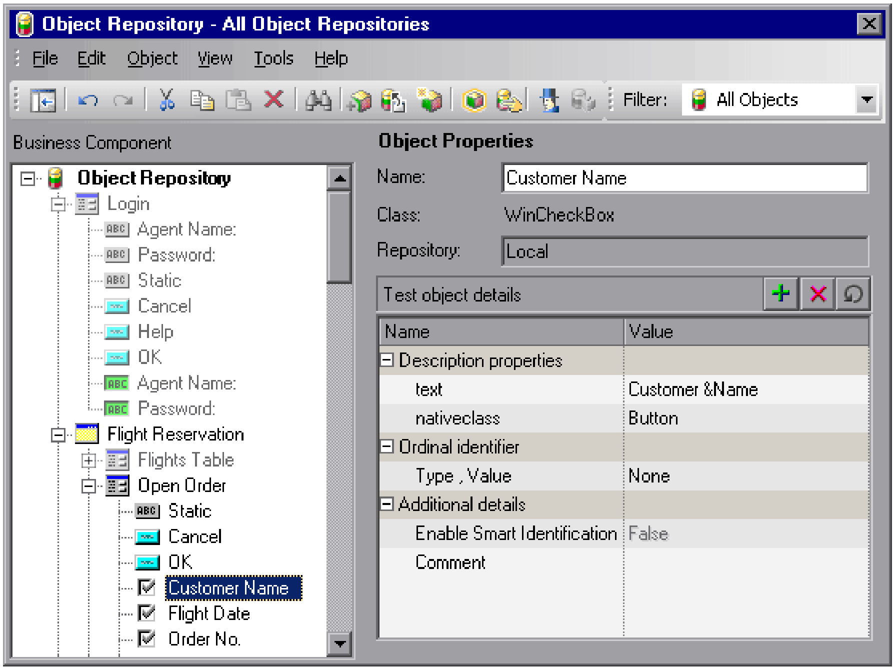


Select an Object

The location you clicked is associated with several objects.
Select the required object from the tree below.

```
Page : Find a Flight: Mercury Tours:
  WebTable : Home
    WebTable : SIGN-OFF
      WebTable : Use our Flight Finder to search for th
        WebTable : Use our Flight Finder to search f
          WebTable :  Flight Details
            Image : findFlights
```

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



Filter: All Objects

Name: Customer Name

Class: WinCheckBox

Repository: Local

Name	Value
⊟ Description properties	
text	Customer &Name
nativeclass	Button
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	False
Comment	

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

Programmatic descriptions enable you to create temporary versions of test objects to 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 could use a programmatic description to add a statement in a user-defined function that enters a value in the new edit box, so that QuickTest can identify the object even though you never recorded on the object or added it to the object repository. For more information on programmatic descriptions, see "Using Programmatic Descriptions" on page 680.

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, refer to the *Mercury QuickTest Professional Object Model Reference*.

Exporting Local Objects to an Object Repository

You can export all of the 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 357). For more information on repository parameters, see Chapter 16, "Managing Object Repositories."

Tip: After you export the local objects, you can use the Object Repository Merge Tool to merge the shared object repository containing the exported objects with another shared object repository. For more information, see Chapter 17, "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. Make sure that the Object Repository window is open.
3. Choose **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 or attachment name, and click **Save** or **OK** (depending on whether you are saving it to the file system or a Quality Center project).

The object repository is exported to the specified shared object repository (a file with a **.tsr** extension). You can now use the new shared object repository like any other shared object repository.

15

Configuring Object Identification

When you record an operation on an object or add an object to the object repository, QuickTest 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.

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 an object you record or add, 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 394.

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 Recording Properties" on page 388
- "Selecting an Ordinal Identifier" on page 394
- "Enabling and Disabling Smart Identification" on page 399
- "Restoring Default Object Identification Settings for Test Objects" on page 400
- "Generating Automation Scripts for Your Object Identification Settings" on page 400

Configuring Mandatory and Assistive Recording 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 you learn 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.

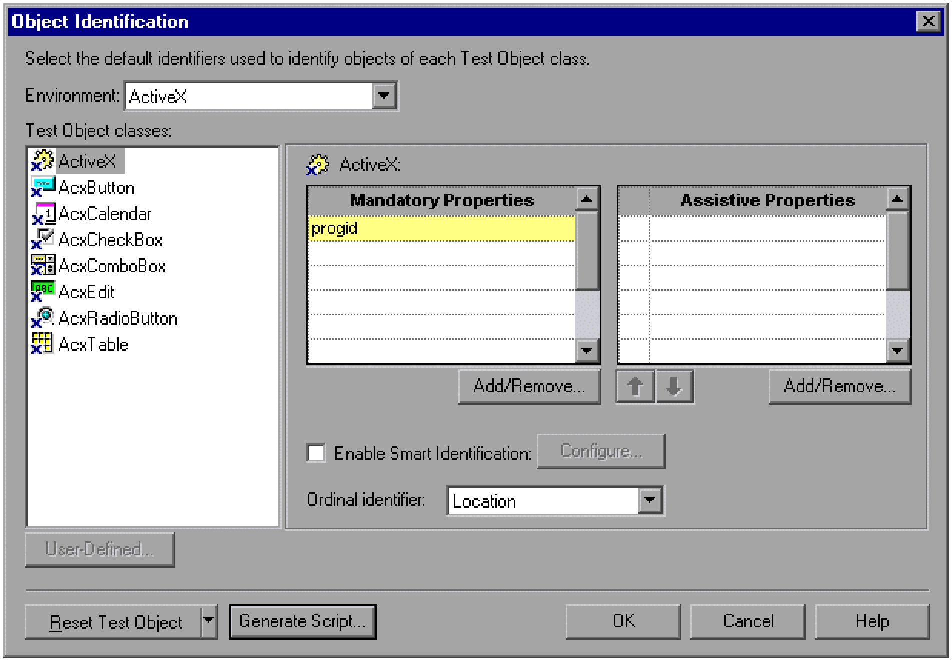
For example, the default mandatory properties for a Web Image object are the **alt**, **html tag**, and **image type** properties. There are no default assistive properties defined. Suppose your Web site contains several space holders for different collections of rotating advertisements. You want to record 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 recorded 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.



Object Identification

Select the default identifiers used to identify objects of each Test Object class.

Environment: ActiveX

Test Object classes:

- ActiveX
- AcxButton
- AcxCalendar
- AcxCheckBox
- AcxComboBox
- AcxEdit
- AcxRadioButton
- AcxTable

ActiveX:

Mandatory Properties
progid

Assistive Properties

□ Enable Smart Identification:

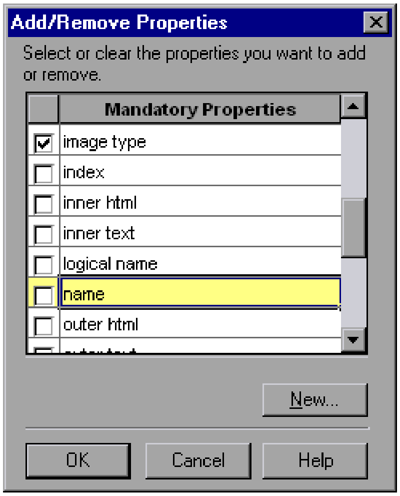
Ordinal identifier: Location

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-in environments. For more information on loading add-ins, see “Loading QuickTest Add-ins” on page 199.

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.



Add/Remove Properties

Select or clear the properties you want to add or remove.

	Mandatory Properties
☑	image type
□	index
□	inner html
□	inner text
□	logical name
□	name
□	outer html

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.

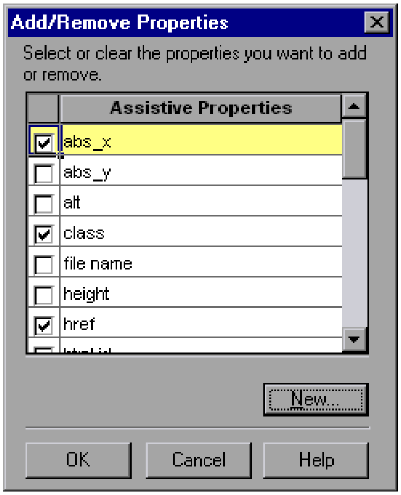
Note: You cannot include the same property in both the mandatory and assistive property lists.

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.



Select or clear the properties you want to add or remove.

	Assistive Properties
☑	abs_x
□	abs_y
□	alt
☑	class
□	file name
□	height
☑	href

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.

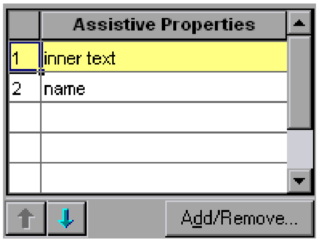
Note: You cannot include the same property in both the mandatory and assistive property lists.

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.



	Assistive Properties
1	inner text
2	name

10 Use the up and down arrows to set your preferred order for the assistive properties. When you learn 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.

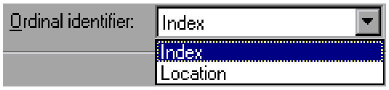
Because the assigned ordinal property value is a relative value and is accurate only in relation to the other objects displayed when QuickTest learns an object, changes in the layout or composition of your application page or screen could cause this value to change, even though the object itself has 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 395.
- **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 396.
- **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 397.

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 14, "Working with Test 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")
```

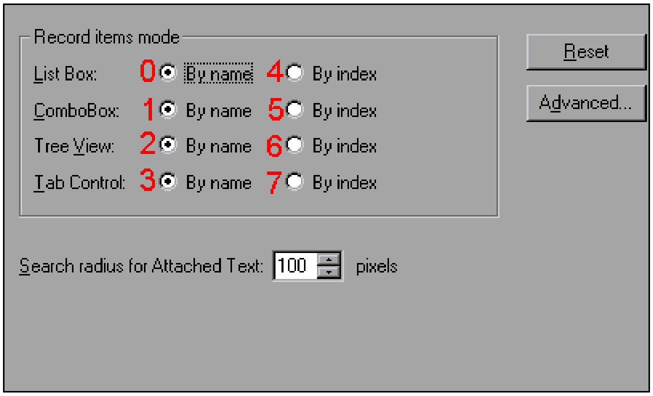
In contrast, the following statement refers to the second item in the list, as that is the first object of any type (WebElement) with the name UserName:

```
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.



Record items mode

List Box: 0 ● By name 4 ○ By index

ComboBox: 1 ● By name 5 ○ By index

Tree View: 2 ● By name 6 ○ By index

Tab Control: 3 ● By name 7 ○ By index

Search radius for Attached Text: 100 pixels

Location property values are object-specific. Therefore, if you use Location:=3 to describe a WinButton test object, 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 14, "Working with Test Objects," and "Defining Run Settings for Your Test" on page 763.Chapter 14, "Working with Test 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 401.

Restoring Default Object Identification Settings for Test Objects

You can restore the default settings for object identification and 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. User-defined objects will be deleted when resetting the Standard Windows environment.

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 709, or refer to 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 Test Results receive a warning status and 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 559.

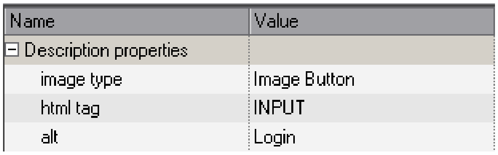
Walking Through a Smart Identification Example

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:



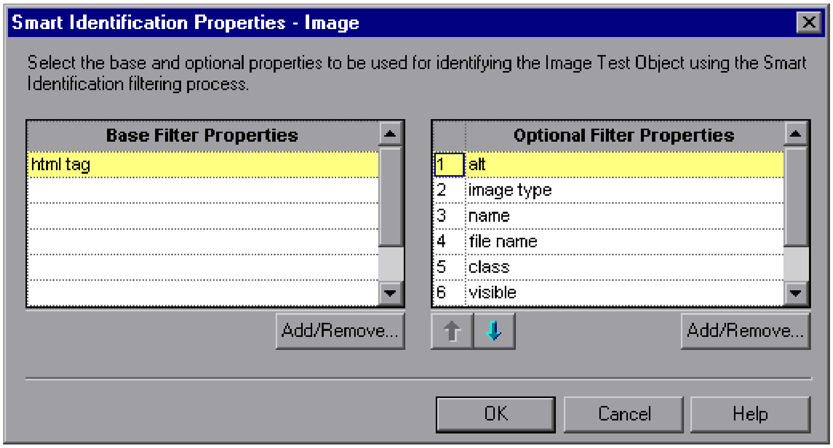
Name	Value
⊟ Description properties	
image type	Image Button
html tag	INPUT
alt	Login

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 when you recorded the click on the Login image:



Smart Identification Properties - Image

Select the base and optional properties to be used for identifying the Image Test Object using the Smart Identification filtering process.

Base Filter Properties
html tag

	Optional Filter Properties
1	alt
2	image type
3	name
4	file name
5	class
6	visible

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 and **image type** = Image Button). 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 **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.

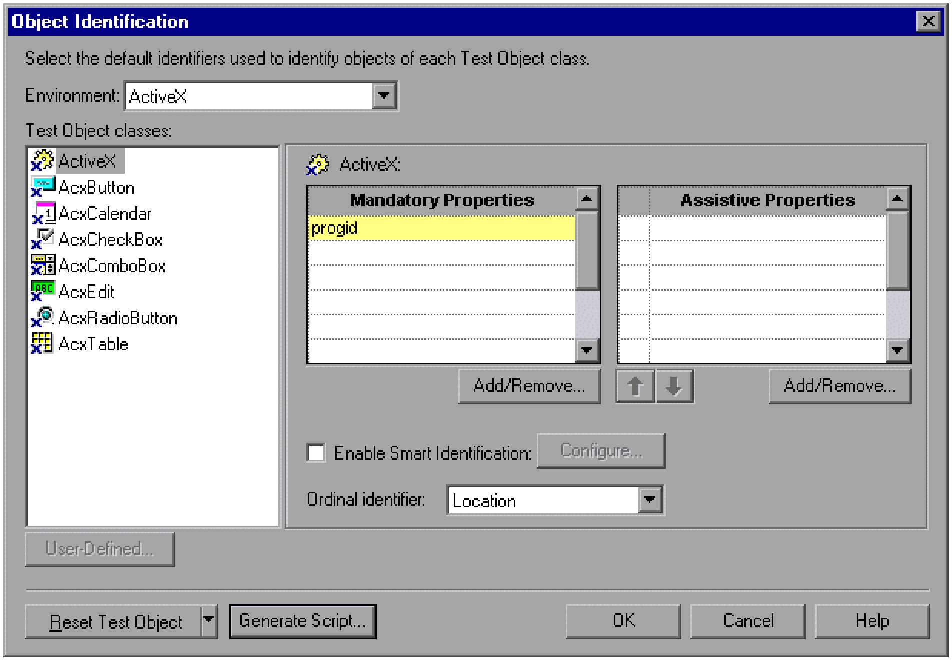
5 QuickTest checks the **file name** property of the two remaining object candidates. Only one of them has the file name login.gif, so QuickTest correctly concludes that it has found the Login button and clicks it.

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.



Object Identification

Select the default identifiers used to identify objects of each Test Object class.

Environment: ActiveX

Test Object classes:

- ActiveX
- AcxButton
- AcxCalendar
- AcxCheckBox
- AcxComboBox
- AcxEdit
- AcxRadioButton
- AcxTable

ActiveX:

Mandatory Properties
progid

Assistive Properties

□ Enable Smart Identification:

Ordinal identifier: Location

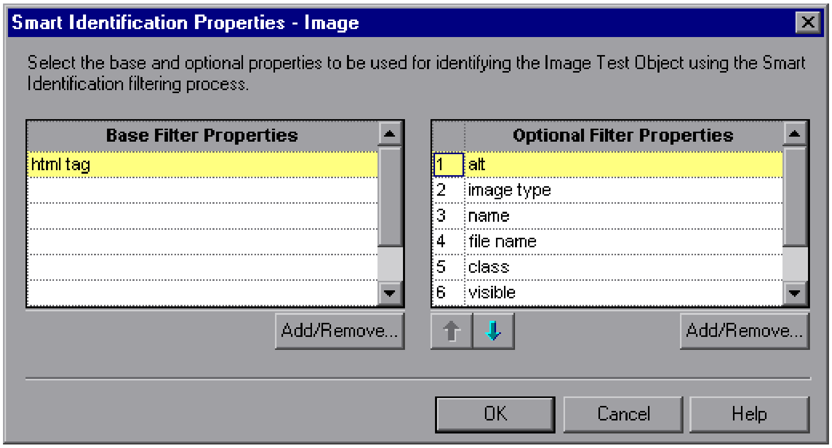
2 Select the appropriate environment in the **Environment** list. The test object classes associated with the selected environment are displayed in the **Test object classes** list.

Note: The environments included in the Environment list are those that correspond to the loaded add-in environments. For more information on loading add-ins, see "Loading QuickTest Add-ins" on page 199.

---

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.



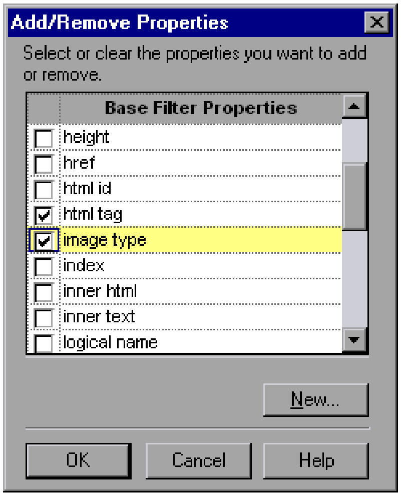
Smart Identification Properties - Image

Select the base and optional properties to be used for identifying the Image Test Object using the Smart Identification filtering process.

Base Filter Properties
html tag

	Optional Filter Properties
1	alt
2	image type
3	name
4	file name
5	class
6	visible

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



Select or clear the properties you want to add or remove.

Base Filter Properties

- □ height
- □ href
- □ html id
- ☑ html tag
- ☑ image type
- □ index
- □ inner html
- □ inner text
- □ logical name

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.

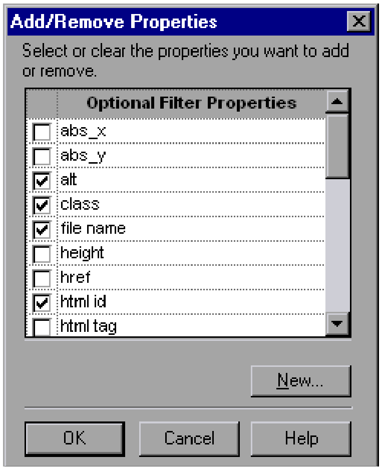
You can specify a new property by clicking **New** and specifying a valid property name in the displayed dialog box.

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.



Add/Remove Properties

Select or clear the properties you want to add or remove.

	Optional Filter Properties
□	abs_x
□	abs_y
☑	alt
☑	class
☑	file name
□	height
□	href
☑	html id
□	html tag

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.

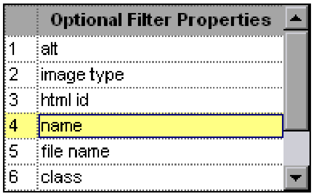
Note: You cannot include the same property in both the base and optional property lists.

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.



	Optional Filter Properties
1	alt
2	image type
3	html id
4	name
5	file name
6	class

↑ ↓

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. You can configure the object identification settings for a user defined object class just as you would any other 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.

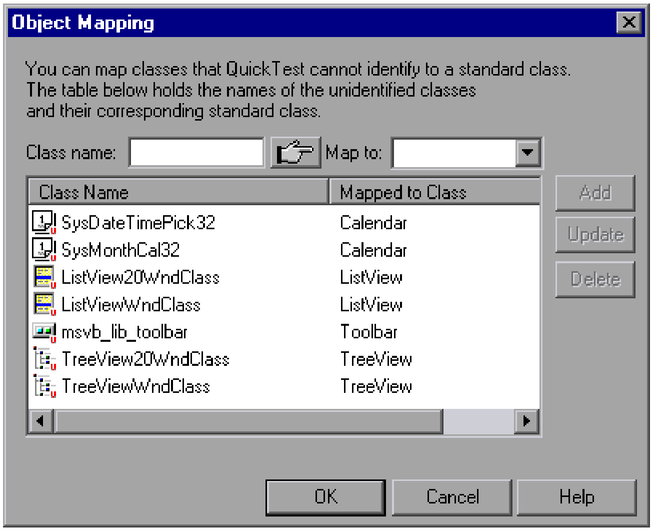
Note: You can define user-defined classes only when **Standard Windows** is selected in the **Environment** box.

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.



You can map classes that QuickTest cannot identify to a standard class. The table below holds the names of the unidentified classes and their corresponding standard class.

Class name:

Map to:

Class Name	Mapped to Class
SysDateTimePick32	Calendar
SysMonthCal32	Calendar
ListView20WndClass	ListView
ListViewWndClass	ListView
msvb_lib_toolbar	Toolbar
TreeView20WndClass	TreeView
TreeViewWndClass	TreeView

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 Recording Properties" on page 388, and "Configuring Smart Identification" on page 401.

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.

16

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.

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 14, "Working with Test 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 88.

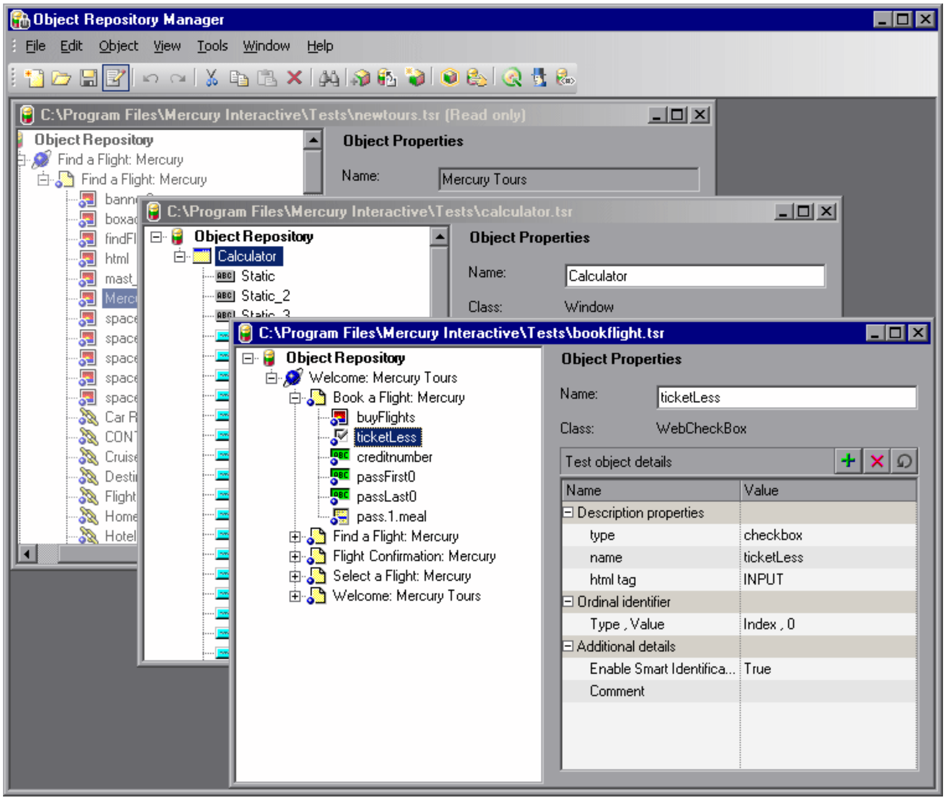
You can use the same shared object repository with multiple components. You can also use multiple object repositories with each component. In addition, you can save objects directly with an component in a local object repository. This enables them to be accessed only from that component.

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 14, "Working with Test 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.



Object Properties

Name: Mercury Tours

Object Properties

Name: Calculator

Class: Window

Object Properties

Name: ticketLess

Class: WebCheckBox

Test object details

Name	Value
⊟ Description properties	
type	checkbox
name	ticketLess
html tag	INPUT
⊟ Ordinal identifier	
Type , Value	Index , 0
⊟ Additional details	
Enable Smart Identifica...	True
Comment	

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 422.

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 430.

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 420.

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 14, “Working with Test 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:

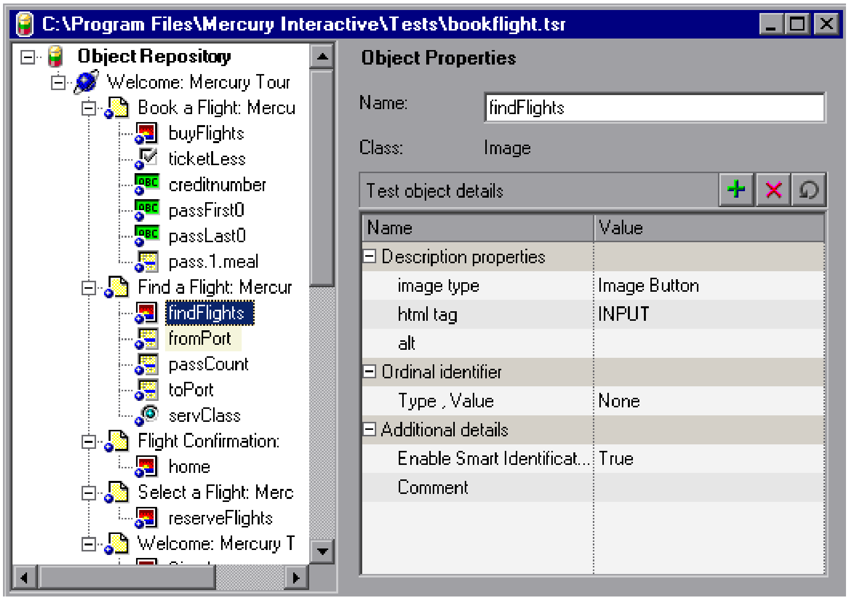
Button	Description
🗋	Enables you to create a new shared object repository. For more information, see “Creating New Object Repositories” on page 424.
📂	Enables you to open a shared object repository from the file system or from Quality Center. For more information, see “Opening Object Repositories” on page 424.
💾	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 426.
📝	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 430.
↶	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 370.
↷	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 370.
✂	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 370.
📋	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 370.

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 370.
✕	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 373.
	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 374.
	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 Objects to the Object Repository” on page 361.
	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 344.
	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 368.
	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 377.
	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 an Object in the Object Repository” on page 378.

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 22.
	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 Using the Object Spy" on page 316.
	Enables you to add, edit, and delete repository parameters in the active shared object repository. For more information, see "Managing Repository Parameters" on page 434.

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 objects in the object repository, together with test object information for the selected object.



C:\Program Files\Mercury Interactive\Tests\bookflight.tsr

```
Object Repository
  Welcome: Mercury Tour
    Book a Flight: Mercu
      buyFlights
      ticketLess
      creditnumber
      passFirst0
      passLast0
      pass.1.meal
    Find a Flight: Mercur
      findFlights
      fromPort
      passCount
      toPort
      servClass
    Flight Confirmation:
      home
    Select a Flight: Merc
      reserveFlights
    Welcome: Mercury T
```

Object Properties

Name: findFlights

Class: Image

Test object details

Name	Value
⊟ Description properties	
image type	Image Button
html tag	INPUT
alt	
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identificat...	True
Comment	

For each test object you select in the tree, the Object Repository window displays information about the selected test object. You can view the test object description of any test object in the shared object repository, modify test objects and their properties, and add objects to the shared object repository. For more information, see “Manipulating Objects in Shared Object Repositories” on page 429 and “Modifying Test Object Details” on page 440.

Each object repository window contains the following information:

Information	Description
Object Repository tree	Contains all test objects in the shared object repository.
Name	Specifies the name that QuickTest assigns to the selected test object. You can change the test object name. For more information, see “Renaming Test Objects” on page 346.
Class	Specifies the class of the selected object.
Test object details	Enables you to view and modify the properties and property values used to identify the selected object during a run session. For more information, see “Modifying Test Object Details” on page 440.

Note: Even when steps containing a test 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 373.

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 88.

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 “Manipulating Objects in Shared Object Repositories” on page 429 and “Saving Object Repositories” on page 426.

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 22.

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 external 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 430.

If the object repository is editable, you can add objects to it, modify it, and save it. For more information, see "Manipulating Objects in Shared Object Repositories" on page 429 and "Saving Object Repositories" on page 426.

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 22.

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.

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). 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.

Manipulating 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. 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 17, "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 430
- "Adding Objects to the Object Repository" on page 361
- "Adding Objects Using the Navigate and Learn Option" on page 431
- "Copying, Pasting, and Moving Objects in the Object Repository" on page 370
- "Deleting Objects from the Object Repository" on page 373

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 Objects Using the Navigate and Learn Option

The **Navigate and Learn** option enables you to add multiple 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 365.

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 Objects to the Object Repository” on page 361.

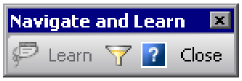
You can use the following keyboard shortcuts when learning objects using the **Navigate and Learn** option:

- **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 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 365.

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 have finished 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 is closed 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 434.

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 195.

Managing Repository Parameters

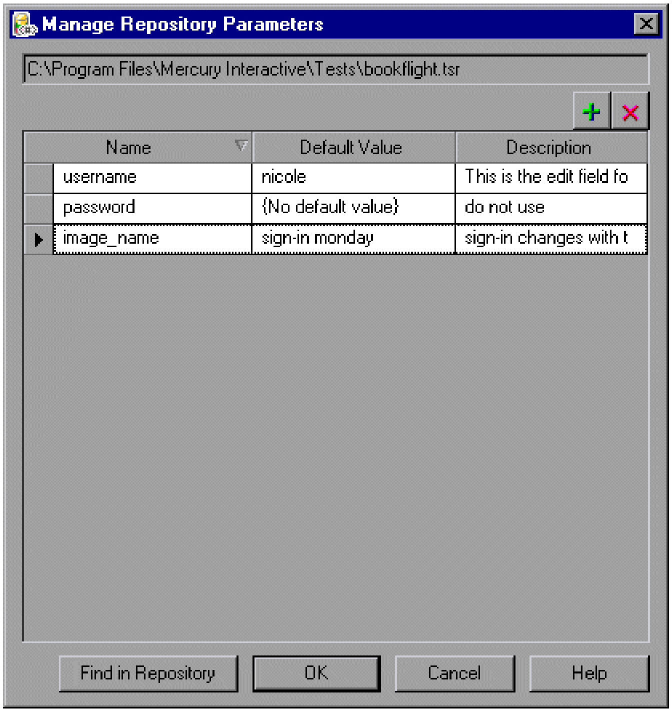
The Manage Repository Parameters dialog box enables you to add, edit, and delete repository parameters for a single shared object repository.

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.



C:\Program Files\Mercury Interactive\Tests\bookflight.tsr

Name	Default Value	Description
username	nicole	This is the edit field fo
password	{No default value}	do not use
image_name	sign-in monday	sign-in changes with t

The Manage Repository Parameters dialog box contains the following information and options:

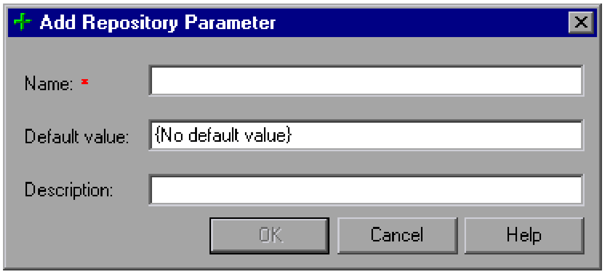
Option	**Description**
Repository name	Displays the name and path of the object repository whose repository parameters you are managing.
✚	Enables you to add a new repository parameter. For more information, see “Adding Repository Parameters” on page 436.
✕	Enables you to delete the currently selected repository parameter(s). For more information, see “Deleting Repository Parameters” on page 439.
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 438.
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.


Add Repository Parameter

Name: *

Default value: {No default value}

Description:

2 In the **Name** box, specify a meaningful name for the parameter. Parameter names must start with an English letter and can contain only alphanumeric characters 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 parameter(s) 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 Test Object Details

The **Test object details** area for shared object repositories open in the Object Repository Manager enables you to view and modify the properties and property values used to identify an object during a run session.

After making sure that your shared object repository is editable, and that it is the active window, you modify test 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 348
- “Defining New Test Object Properties” on page 352
- “Updating Test Object Properties from an Object in Your Application” on page 344
- “Restoring Default Properties for a Test Object” on page 346
- “Removing Properties from a Test Object Description” on page 354
- “Specifying Ordinal Identifiers” on page 355
- “Renaming Test Objects” on page 346

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 test 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 441.

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 433.

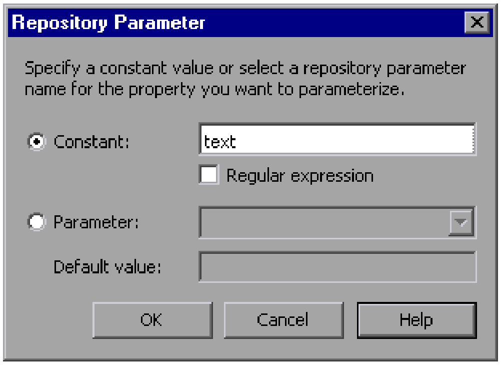
To specify a property value:

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.

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.


Repository Parameter

Specify a constant value or select a repository parameter name for the property you want to parameterize.

● Constant: text

□ Regular expression

○ Parameter:

Default value:

4 Choose one of the following options to specify a value for the property:

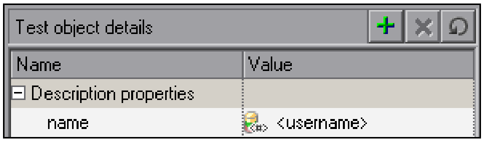
➤ Select the **Constant** radio button and specify a constant value. You can also enter a constant value directly in the value cell of the **Test object details** area. If you used a regular expression in the constant value, select the **Regular expression** check box.

➤ 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 434.

---

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.



Test object details

Name	Value
⊟ Description properties	
name	<username>

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 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 374
- "Highlighting an Object in Your Application" on page 377
- "Locating an Object in the Object Repository" on page 378

Performing Merge Operations

The Object Repository Merge Tool enables you to merge 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 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 17, "Merging Shared Object Repositories."

Note: While the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager.

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 22.

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 447. 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, refer to 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 446.

To import from XML:

1 Choose **File** > **Import from XML**. The Import from XML 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 Import from XML dialog box.

---

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 contents of an object repository to an XML file. This enables you to easily 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 you want to export is the active window.

2 Make sure that the object repository you want to export is saved.

3 Choose **File** > **Export to XML**. The Export to XML 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 to XML dialog box.

---

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 object repository is 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**).

Manipulating Object Repositories Using Automation

QuickTest provides an Object Repository automation object model that enables you to manipulate 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 manipulate 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, refer to the *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 709.

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, refer to the documentation included with your development environment or to general documentation for the programming language.

17

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.

About Merging Shared Object Repositories

QuickTest Professional provides the ability to merge existing assets from two object repositories into a single shared object repository using the Object Repository Merge Tool. This tool enables you to 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 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 suggested resolution for each conflict, or modify each conflict resolution individually, according to your requirements.

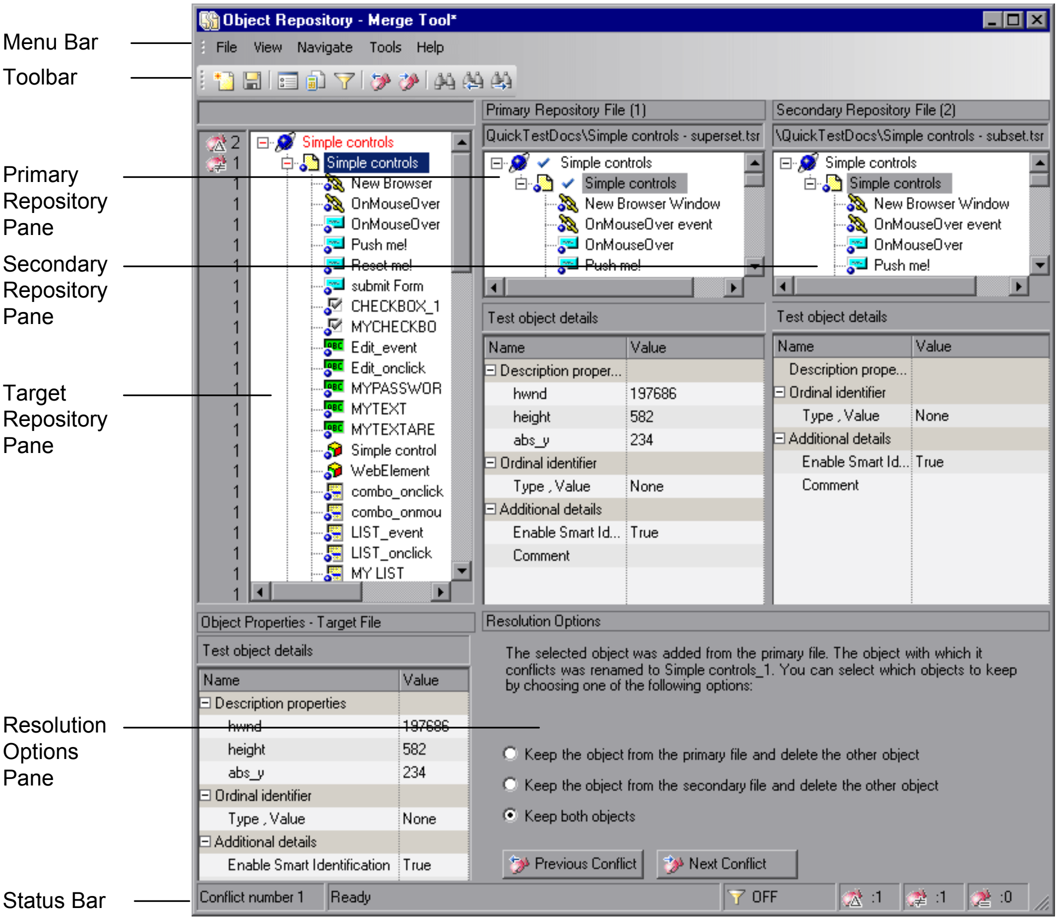
The Object Repository Merge Tool also enables you to merge objects from the local object repository of one or more actions into a shared object repository. For example, if QuickTest learned objects locally in a specific action in your test, you may want to add the objects to the shared object repository, so that they are available to all actions in different tests that use that object repository.

Note: When the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager. For more information on the Object Repository Manager, see Chapter 16, “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:



QuickTestDocs\Simple controls - superset.tsr

\QuickTestDocs\Simple controls - subset.tsr

Target tree: Simple controls (2); Simple controls (1); New Browser, OnMouseOver, OnMouseOver, Push me!, Reset me!, submit Form, CHECKBOX_1, MYCHECKBO, Edit_event, Edit_onclick, MYPASSWOR, MYTEXT, MYTEXTARE, Simple control, WebElement, combo_onclick, combo_onmou, LIST_event, LIST_onclick, MY LIST (1 each)

Primary tree: Simple controls ✓; Simple controls ✓; New Browser Window, OnMouseOver event, OnMouseOver, Push me!

Secondary tree: Simple controls; Simple controls; New Browser Window, OnMouseOver event, OnMouseOver, Push me!

Test object details (primary)

Name	Value
⊟ Description proper...	
hwnd	197686
height	582
abs_y	234
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Id...	True
Comment	

Test object details (secondary)

Name	Value
Description prope...	
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Id...	True
Comment	

Test object details (target)

Name	Value
⊟ Description properties	
hwnd	197686
height	582
abs_y	234
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	True

The selected object was added from the primary file. The object with which it conflicts was renamed to Simple controls_1. You can select which objects to keep by choosing one of the following options:

○ Keep the object from the primary file and delete the other object

○ Keep the object from the secondary file and delete the other object

● Keep both objects

Conflict number 1 | Ready | OFF | :1 | :1 | :0

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 "Performing Commands Using Shortcut Keys" on page 460.
- **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 459.
- **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 455.
- **Primary Repository Pane.** Displays the objects in the primary object repository. For more information, see "Primary and Secondary Repository Panes" on page 457.
- **Secondary Repository Pane.** Displays the objects in the secondary object repository. For more information, see "Primary and Secondary Repository Panes" on page 457.
- **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 457.
- **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 458.

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 464.

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 481.
- 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 conflict number (if any) of the object selected in the target object repository pane, the filter status, and a legend of the icons used in the target object repository pane.

- 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.
- The following icons may be displayed in the status bar and (and in the target object repository pane):
  - 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 475.

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 474.

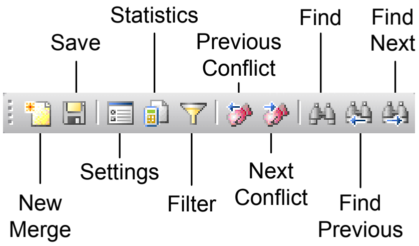
Click the **Filter** icon in the status bar to view the Filter dialog box. The filter is shown as **ON** in the status bar when a filter is currently in use. For more information, see "Filtering the Target Repository Pane" on page 480.

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, as described in "Performing Commands Using Shortcut Keys" on page 460. 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 toolbar.



Performing Commands Using Shortcut Keys

You can perform some Object Repository Merge Tool commands by pressing shortcut keys. The shortcut keys listed below are shown next to the respective menu commands.

You can perform the following **File** menu commands by pressing the corresponding shortcut keys:

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.

You can perform the following **Navigate** menu commands by pressing the corresponding shortcut keys:

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.

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.

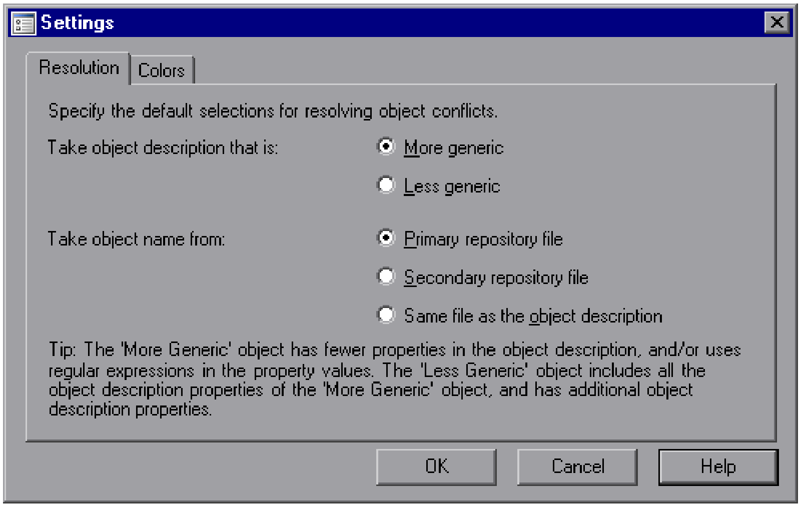
Specifying Default Resolution Settings

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.



Specify the default selections for resolving object conflicts.

Take object description that is:
● More generic
○ Less generic

Take object name from:
● Primary repository file
○ Secondary repository file
○ Same file as the object description

Tip: The 'More Generic' object has fewer properties in the object description, and/or uses regular expressions in the property values. The 'Less Generic' object includes all the object description properties of the 'More Generic' object, and has additional object description properties.

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.

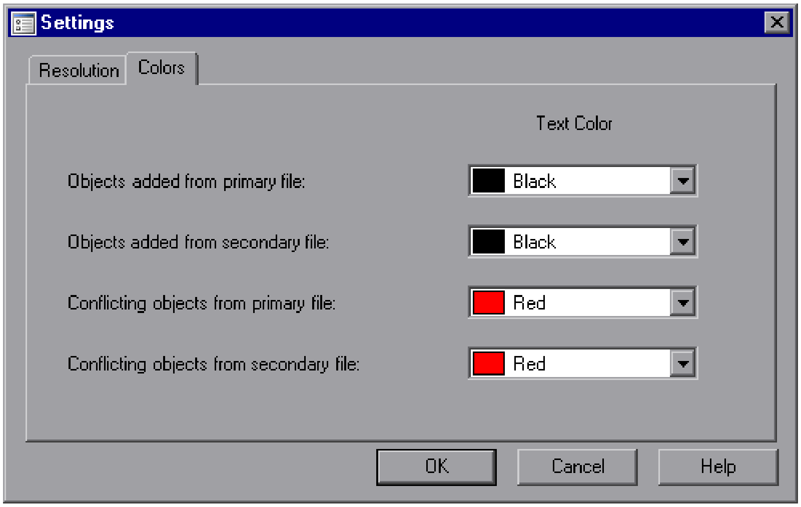
Specifying Color Settings

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 more easily the status of each object.

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.



Text Color

Objects added from primary file: ■ Black

Objects added from secondary file: ■ Black

Conflicting objects from primary file: 🟥 Red

Conflicting objects from secondary file: 🟥 Red

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.

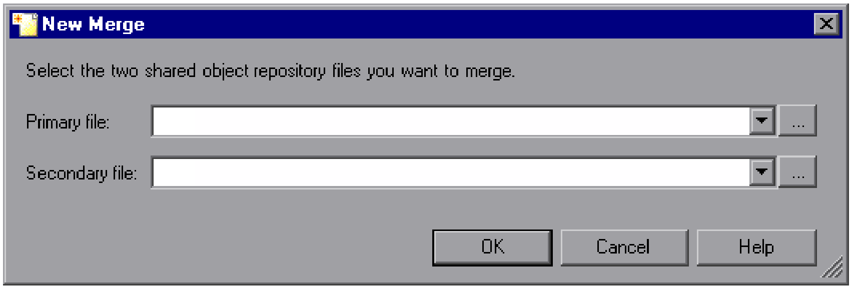
Merging Two Object Repositories

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.



Select the two shared object repository files you want to merge.

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 461, and then perform the merge.

2 In the **Primary file** and **Secondary file** boxes, enter or browse to and select 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.

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 474, 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 478.
- Filter the objects in the target object repository, as described in “Filtering the Target Repository Pane” on page 480.
- 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 483.

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 461.

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.

Note: You can only merge local object repositories 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 16, "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.

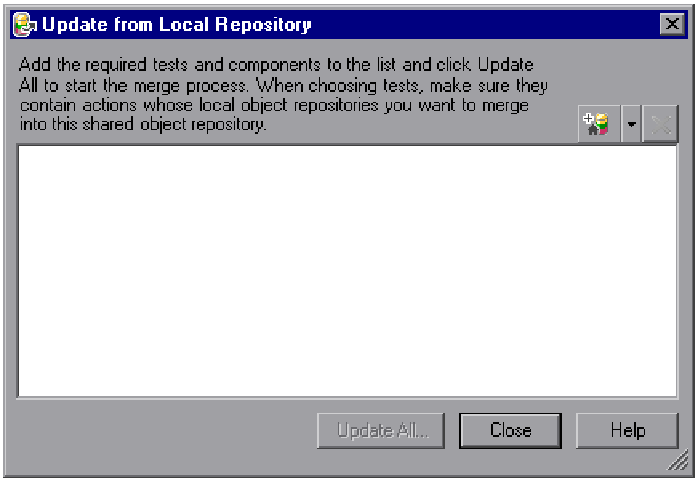
If you are currently connected to a Quality Center project, the Open Shared Object Repository dialog box displays the component tree for the project. Select a component to view the shared object repositories attached to the component.

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.


Update from Local Repository

Add the required tests and components to the list and click Update All to start the merge process. When choosing tests, make sure they contain actions whose local object repositories you want to merge into this shared object repository.

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 step 5 to add additional components if required.

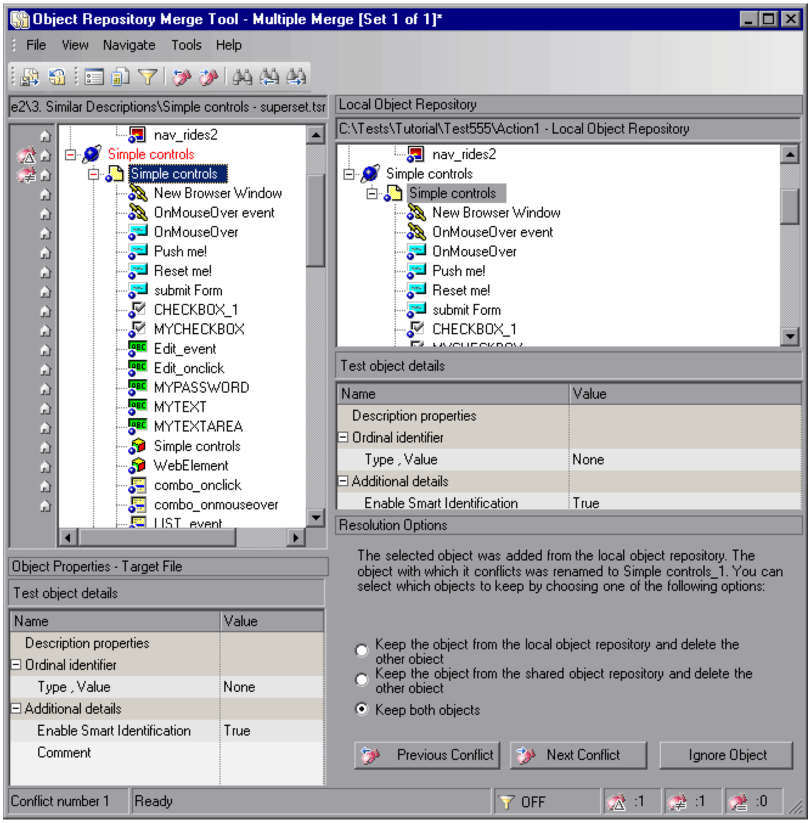
Note: The local object repositories associated with all the components are included in the merge. If you want to remove an component from the merge, select it in the list and click **Delete**.

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 474, 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.



Test object details

Name	Value
Description properties	
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	True

Test object details

Name	Value
Description properties	
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	True
Comment	

Resolution Options

The selected object was added from the local object repository. The object with which it conflicts was renamed to Simple controls_1. You can select which objects to keep by choosing one of the following options:

○ Keep the object from the local object repository and delete the other object
○ Keep the object from the shared object repository and delete the other object
● Keep both objects

Conflict number 1 | Ready

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 478.

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).

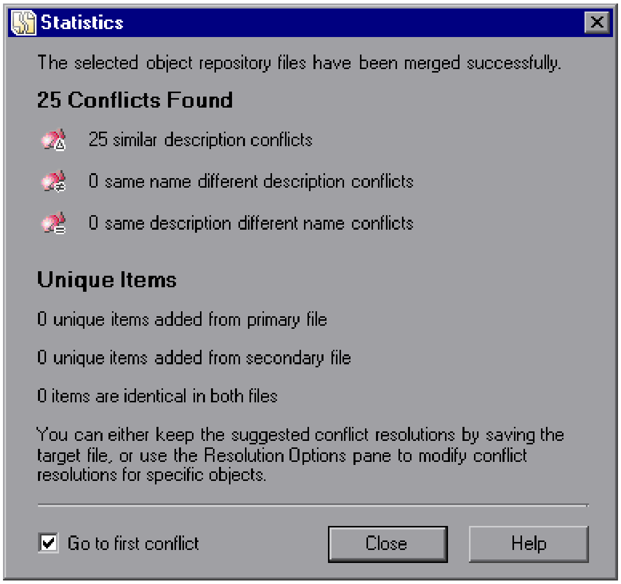
11 Click **Yes** to save your changes between merges. If you click **No**, the current merge (objects merged from the last component) will not be saved.

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.



Statistics

The selected object repository files have been merged successfully.

25 Conflicts Found

25 similar description conflicts

0 same name different description conflicts

0 same description different name conflicts

Unique Items

0 unique items added from primary file

0 unique items added from secondary file

0 items are identical in both files

You can either keep the suggested conflict resolutions by saving the target file, or use the Resolution Options pane to modify conflict resolutions for specific objects.

☑ Go to first conflict

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 478.
- 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 476.

  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 461.

- **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 477.

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 461.

- **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 477.

  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 461.

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:

- Taking the object description from the object that is added from the primary repository.
- Taking the object description from the object that is added from the secondary object repository.
- Taking both objects into the target object repository. 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.
- (When updating a shared object repository from a local object repository) Ignoring the object from the local object repository and keeping the object from the shared 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 object 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.
- (When updating a shared object repository from a local object repository) Ignoring the object from the local object repository and keeping the object from the shared 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.
- (When updating a shared object repository from a local object repository) Ignoring the object from the local object repository and keeping the object from the shared 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 461.

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 461.

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 475.

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 483.

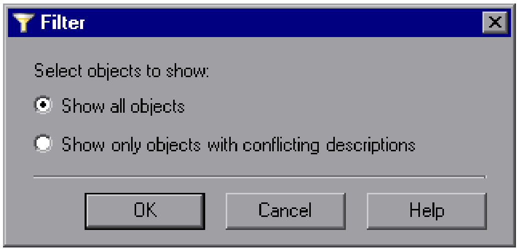
Filtering the Target Repository Pane

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.



Select objects to show:

● Show all objects

○ Show only objects with conflicting descriptions

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 had description conflicts

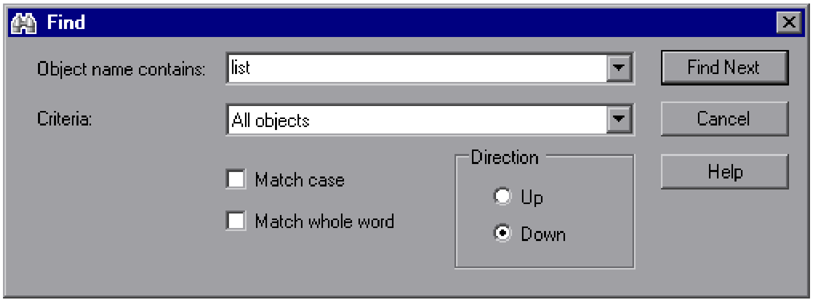
3 Click **OK**. The objects in the pane are filtered and the target object repository displays only the requested object types. A progress bar is displayed in the status bar during the filter process.

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.



Object name contains: list

Criteria: All objects

□ Match case

□ Match whole word

Direction

○ Up

● Down

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 type(s) 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 Navigate 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.

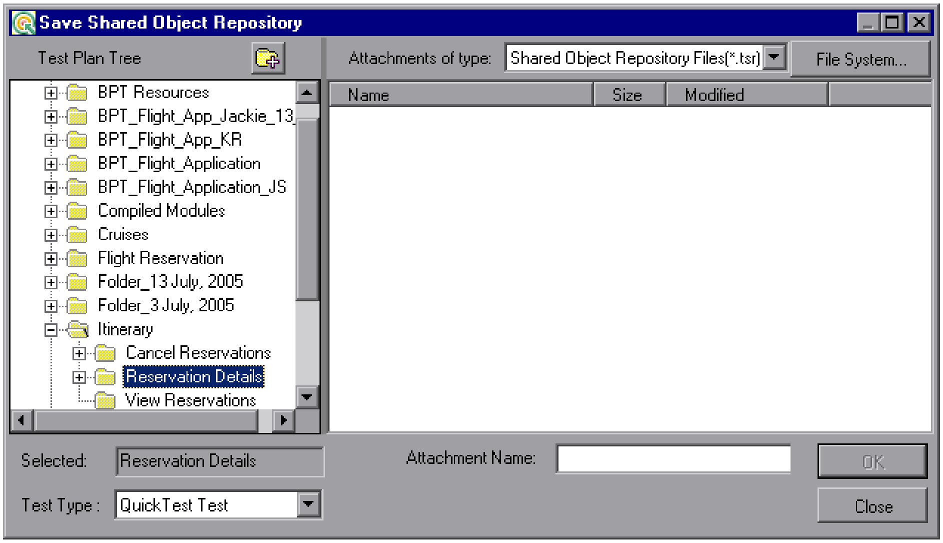
Saving the Object Repository to a Quality Center Project

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 area(s) so that the objects in the object repository can be accessed by components. For more information, see "Managing Shared Object Repositories" on page 88.

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.



Attachments of type: Shared Object Repository Files(*.tsr)

Selected: Reservation Details

Test Type : QuickTest Test

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.

18

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.

About Comparing Shared Object Repositories

QuickTest Professional provides the ability 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 494.

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.

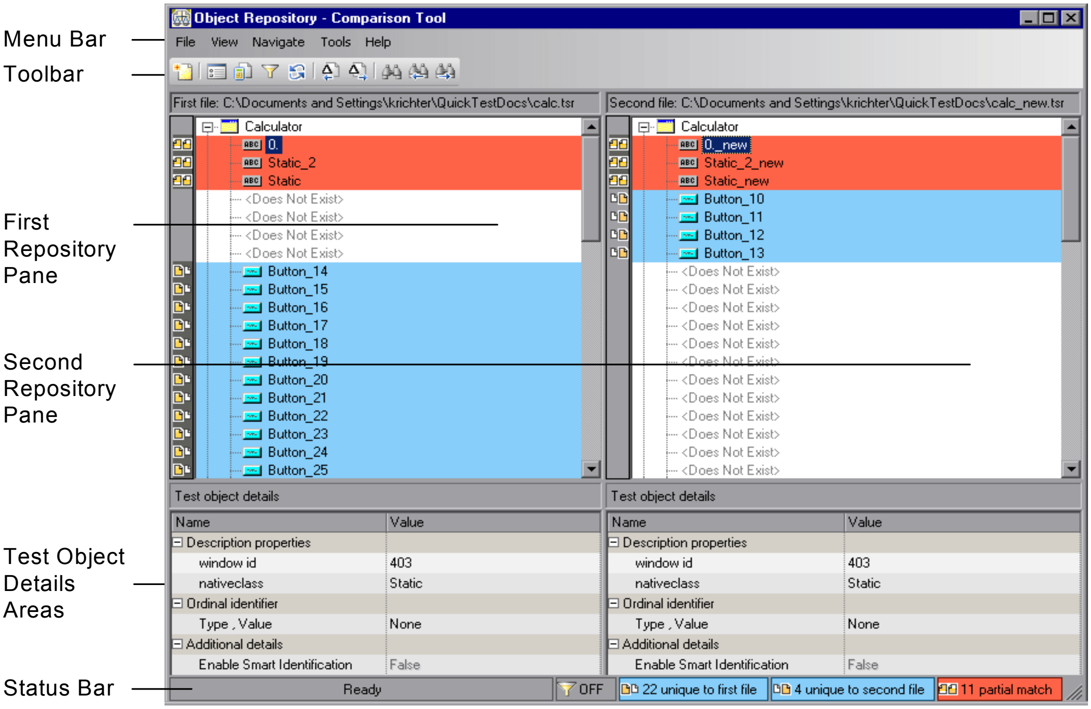
You can use the information displayed by the Object Repository Comparison Tool when managing or merging object repositories. For more information, see Chapter 16, "Managing Object Repositories," or Chapter 17, "Merging Shared Object Repositories."

Tip: You can 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:



First file: C:\Documents and Settings\krichter\QuickTestDocs\calc.tsr

- Calculator
  - 0.
  - Static_2
  - Static
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - Button_14
  - Button_15
  - Button_16
  - Button_17
  - Button_18
  - Button_19
  - Button_20
  - Button_21
  - Button_22
  - Button_23
  - Button_24
  - Button_25

Second file: C:\Documents and Settings\krichter\QuickTestDocs\calc_new.tsr

- Calculator
  - 0._new
  - Static_2_new
  - Static_new
  - Button_10
  - Button_11
  - Button_12
  - Button_13
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>
  - <Does Not Exist>

Test object details

Name	Value
⊟ Description properties	
window id	403
nativeclass	Static
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	False

Test object details

Name	Value
⊟ Description properties	
window id	403
nativeclass	Static
⊟ Ordinal identifier	
Type , Value	None
⊟ Additional details	
Enable Smart Identification	False

Ready | OFF | 22 unique to first file | 4 unique to second file | 11 partial match

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 Commands Using Shortcut Keys" on page 493.

- **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 493.
- **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 490.
- **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 490.
- **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 492.

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.

Note: 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 495.

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 494.

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 Commands Using Shortcut Keys" on page 493.

- 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 502.
- 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 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 500.
- 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

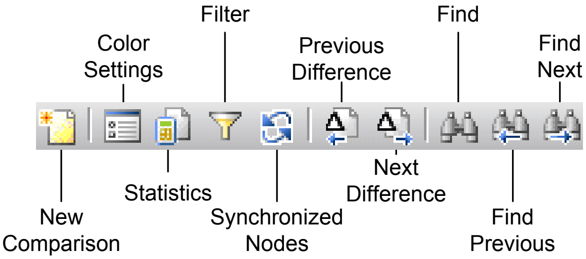
For more information on all difference types, see "Understanding Object Differences" on page 494.

Using Object Repository Comparison Tool Commands

You can select Object Repository Comparison Tool commands from the menu bar or from the toolbar. You can perform certain commands by pressing shortcut keys, as described in "Performing Commands Using Shortcut Keys" on page 493.

Using Toolbar Commands

You can perform frequently used commands by clicking buttons in the toolbar.



Performing Commands Using Shortcut Keys

You can perform some Object Repository Comparison Tool commands by pressing shortcut keys.

You can open context-sensitive Help by pressing the F1 key.

The shortcut keys listed below are shown next to the respective menu commands.

You can perform the following **File** menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
New Comparison	CTRL+N	Enables you to specify two object repositories on which to perform a new comparison operation.

You can perform the following **Navigate** menu commands by pressing the corresponding shortcut keys:

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.
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.

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.

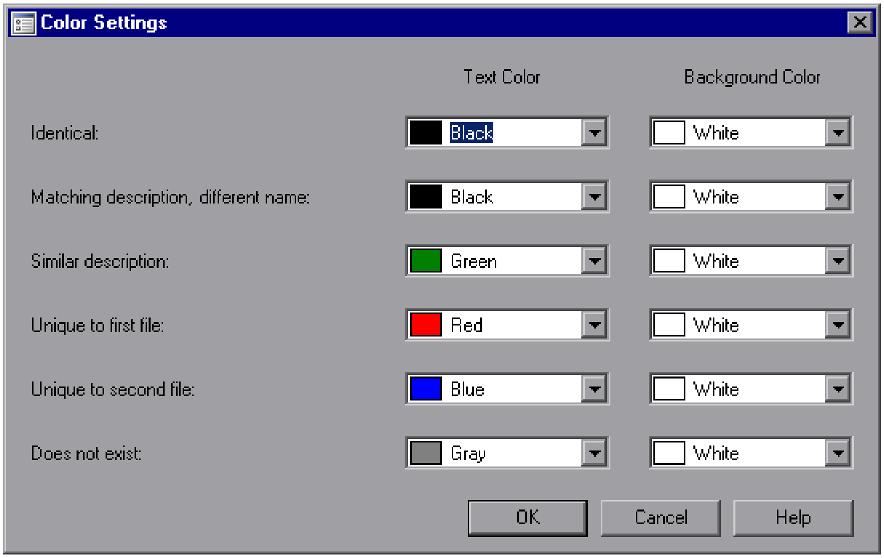
Changing Color Settings

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.



Color Settings

	Text Color	Background Color
Identical:	Black	White
Matching description, different name:	Black	White
Similar description:	Green	White
Unique to first file:	Red	White
Unique to second file:	Blue	White
Does not exist:	Gray	White

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.

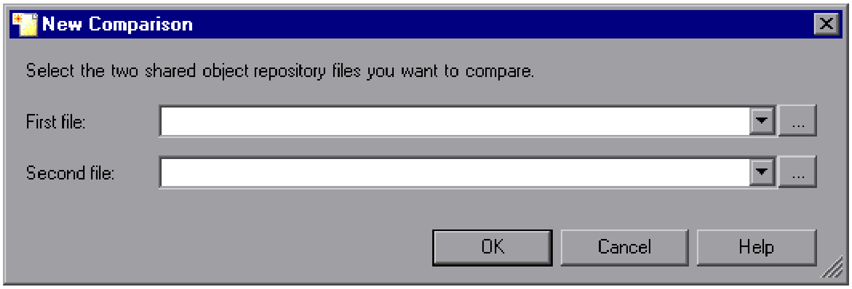
Comparing Object Repositories

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.



New Comparison
Select the two shared object repository files you want to compare.
First file:
Second file:
OK
Cancel
Help

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 495, 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.

---

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.

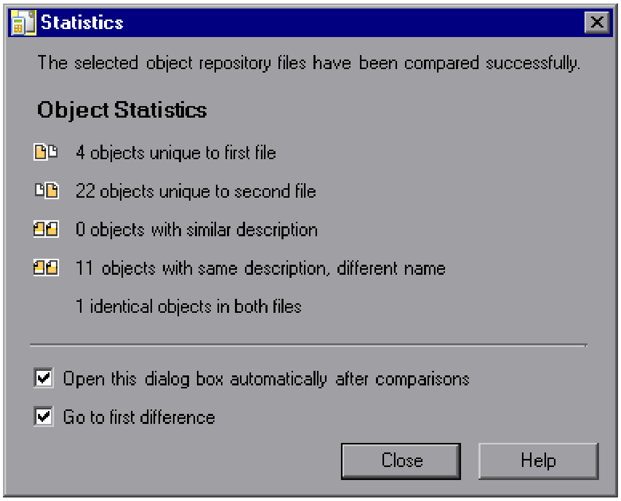
5 Review the statistics, as described in “Viewing Comparison Statistics” on page 499, and click **Close**.

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 500.
- ➤ Find specific objects in the object repositories, as described in “Finding Specific Objects” on page 502.

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.



Statistics

The selected object repository files have been compared successfully.

Object Statistics

4 objects unique to first file

22 objects unique to second file

0 objects with similar description

11 objects with same description, different name

1 identical objects in both files

☑ Open this dialog box automatically after comparisons

☑ Go to first difference

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 494.
- 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 490.

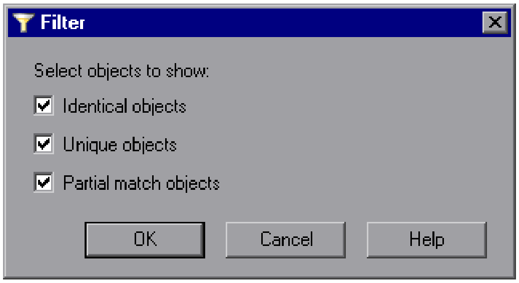
Tip: Select the **Go to first difference** check box to jump to the first difference in the object repositories immediately after you close the Statistics dialog box.

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.



Select objects to show:

☑ Identical objects

☑ Unique objects

☑ Partial match objects

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.

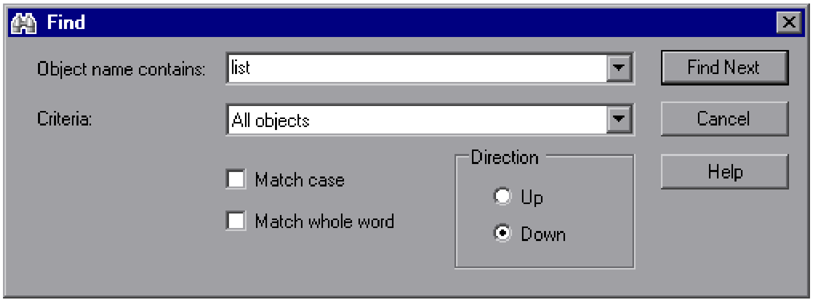
Tip: The **Synchronized Nodes** button in the toolbar is surrounded by a border when the object repositories are currently synchronized. Click the **Synchronized Nodes** button again to navigate the two object repositories independently. When the object repositories are synchronized, you can also press 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.



Object name contains: list

Criteria: All objects

□ Match case

□ Match whole word

Direction

○ Up

● Down

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 V

Running and Debugging Components

19

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 Mercury products to run tests and components** check box (from QuickTest, choose **Tools** > **Options** > **Run**).

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 **Start 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, refer to "Running Part of Your Component" in the *QuickTest Professional User's Guide*.

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 631. 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 220.

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 73.)
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 83.)
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 517.
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 Start from Step Commands

In addition to stepping into, out of, and over a step while debugging, you can use the **Run to Step** and **Start 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

➤ 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 step 2 in the "Start from Step" section, below.

Start 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 or Web site 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.

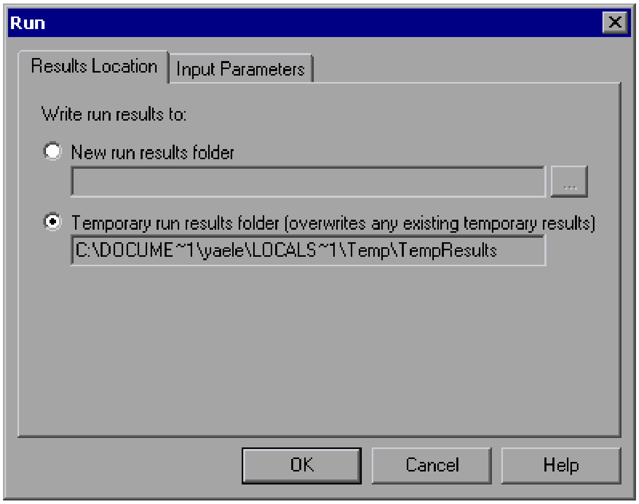
To instruct QuickTest to run from a particular step:

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** > **Start 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.



Run

Results Location | Input Parameters

Write run results to:

○ New run results folder

● Temporary run results folder (overwrites any existing temporary results)
C:\DOCUME~1\yaele\LOCALS~1\Temp\TempResults

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 530, and “Understanding the Input Parameters Tab” on page 531.

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 21, “Analyzing Test 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 9, “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 site or 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
- ➤ 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 518.

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:

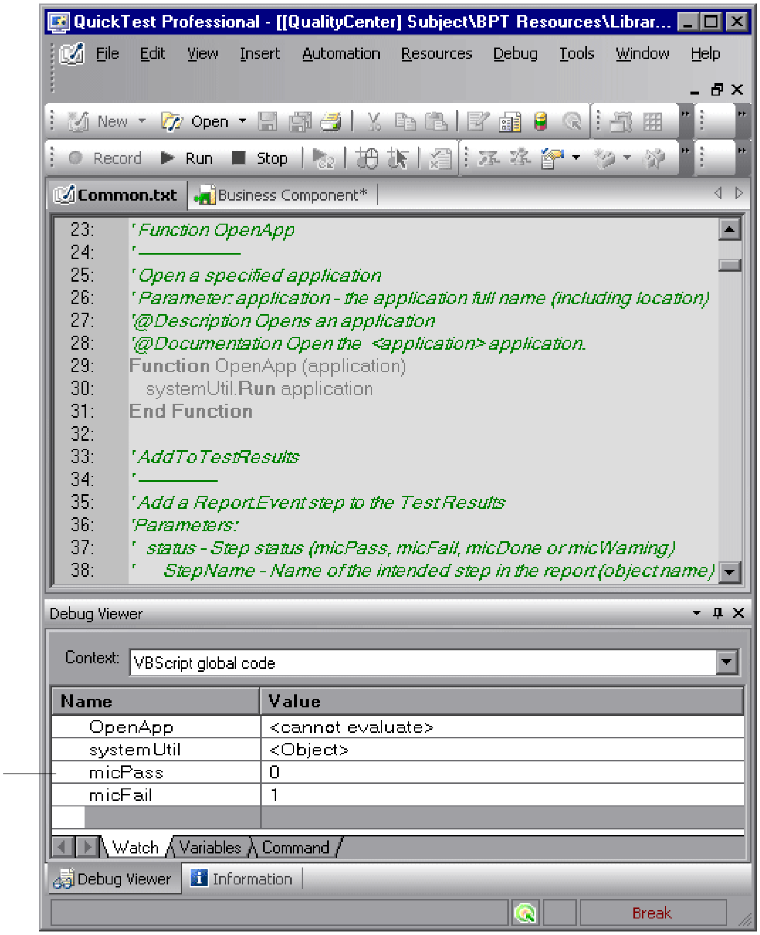
Click the **Clear All Breakpoints** button, or choose **Debug** > **Clear All Breakpoints**. All breakpoint symbols are removed from the left margin of the QuickTest window.

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.



```
'Function OpenApp
'__________
'Open a specified application
'Parameter: application - the application full name (including location)
'@Description Opens an application
'@Documentation Open the <application> application.
Function OpenApp (application)
  systemUtil.Run application
End Function

'AddToTestResults
'__________
'Add a ReportEvent step to the Test Results
'Parameters:
'  status - Step status (micPass, micFail, micDone or micWarning)
'    StepName - Name of the intended step in the report (object name)
```

Context: VBScript global code

Name	Value
OpenApp	<cannot evaluate>
systemUtil	<Object>
micPass	0
micFail	1

Debug Viewer

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

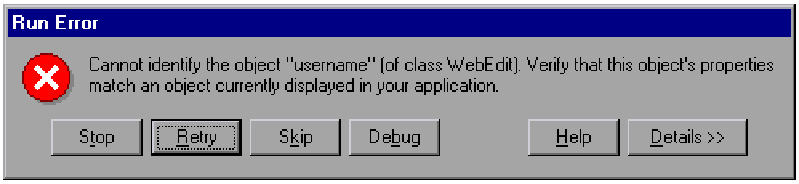
QuickTest automatically displays the current value of all variables in the current function in the Variables tab—up to the point where the function library is stopped or paused. For example, if you are stepping through a function, as you step into each step, QuickTest adds the current value for any step 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

The Run Error message box displayed during a run session offers a number of buttons for dealing with errors encountered:



Run Error

Cannot identify the object "username" (of class WebEdit). Verify that this object's properties match an object currently displayed in your application.

- **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.

- **Details.** Expands the message box to display additional information on the error.

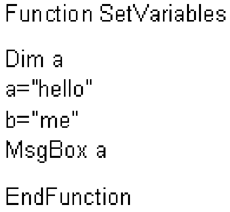
Practicing Debugging a Function

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 23, "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 517.

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**.

20

Running Components

After you create a component, you can run it to check the behavior of your application.

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 21, “Analyzing Test 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 14, “Working with Test 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 532.

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:

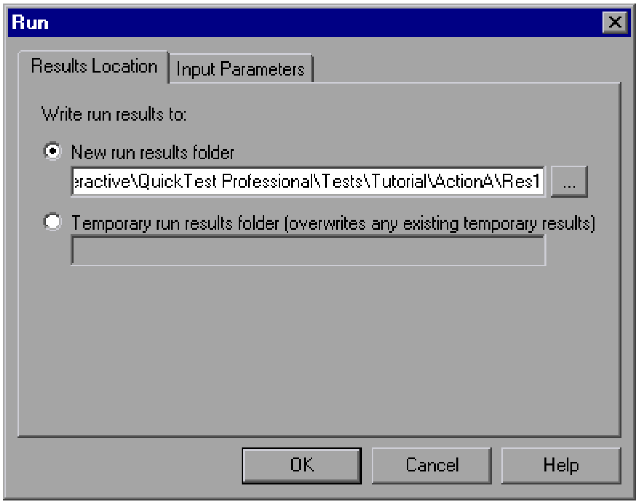
Open ▾

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.

▶ Run

2 Click the **Run** button on the toolbar, or choose **Automation** > **Run**. The Run dialog box opens.



Write run results to:

● New run results folder

eractive\QuickTest Professional\Tests\Tutorial\ActionA\Res1

○ Temporary run results folder (overwrites any existing temporary results)

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 530, and "Understanding the Input Parameters Tab" on page 531.
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 21, "Analyzing Test 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 9, "Setting Global Testing Options."

Tip: If you want to interrupt a run session, do either of the following:

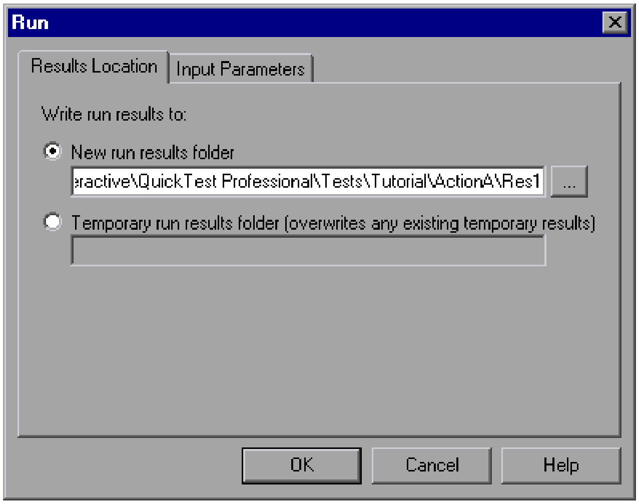
⏸ Click the **Pause** button in the Debug toolbar or choose **Debug** > **Pause**. The run pauses. To resume running a paused run session, click the **Run** button or choose **Automation** > **Run**.

■ Stop 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.



Write run results to:

● New run results folder

eractive\QuickTest Professional\Tests\Tutorial\ActionA\Res1

○ Temporary run results folder (overwrites any existing temporary 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.

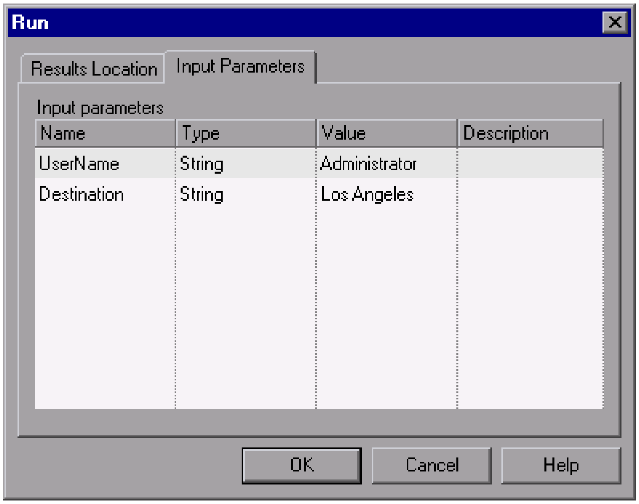
  Accept the default settings, or enter a new path by typing it in the text box or clicking the browse button to locate a different folder. The folder must be new, empty, or contain only QuickTest test or component files.

- **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.



Input parameters

Name	Type	Value	Description
UserName	String	Administrator	
Destination	String	Los Angeles	

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 263. For more information on using parameters, see “Working with Parameters” on page 170.

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 Start from Step Commands” on page 513.

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 530, and “Understanding the Input Parameters Tab” on page 531.
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 21, "Analyzing Test 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 9, "Setting Global Testing Options."

Updating a Component

When you update a component, QuickTest runs the component to update the test object descriptions. You can choose to update the data for an entire component or only part of it, according to the run and debug run options you select. After you save the component, the updated data is used for subsequent runs.

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 16, "Managing Object Repositories."

To run a component to update the test object descriptions:

1. If your component is not already open, choose **File** > **Open** > **Business/Scripted Component**, or click the **Open** arrow and choose **Business/Scripted Component**.
2. Select a component and click **Open**. The component opens and the title bar displays the component name.

Tip: If you recently opened your component, you can also choose it from the recent files list in the **File** menu.

---

3. Choose **Automation** > **Update Run Mode**, or click the **Update Run Mode** button. The **Update Run Mode** menu option and button are toggled, and your component will be updated according to the type of run you select in the following step.
4. Select the type of run to perform from the following options:

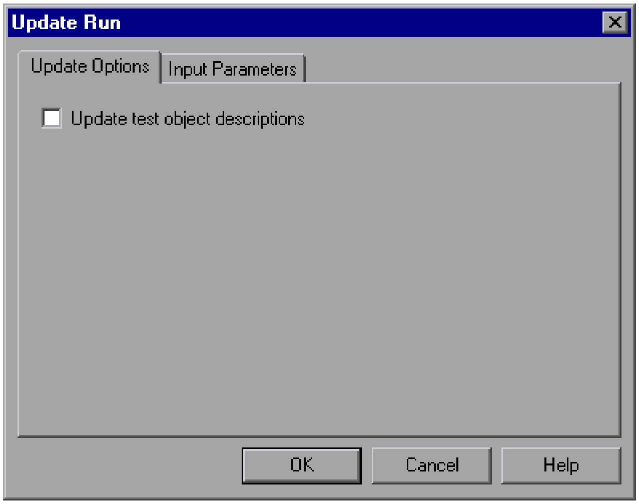
   - **Automation** > **Run.** Runs and updates your entire component. You can also click the **Run** button.
   - **Automation** > **Run from Step.** Runs and updates your component from the selected step to the end of the component.

   You can also select one of the following debug run modes to debug and update your component:

   - **Debug** > **Step Into** or click the **Step Into** button. Debugs and updates only the current step of the component.
   - **Debug** > **Step Over** or click the **Step Over** button. Debugs and updates only the current step of the component.
   - **Debug** > **Step Out** or click the **Step Out** button. Debugs and updates the current step of the component.
   - **Debug** > **Run to Step.** Debugs and updates the component from the current location in the component until it reaches the selected step.
   - **Debug** > **Start from Step.** Debugs and updates the component from the selected step to the end of the component.

For more information on debugging, see Chapter 19, "Debugging Components and Function Libraries."

The Update Run dialog box opens.



□ Update test object descriptions

5 Specify the settings for the update run process. For more information, see "Understanding the Update Options Tab" on page 537, and "Understanding the Input Parameters Tab" on page 531.

Note: The run results for an update run session are always saved in a temporary location.

6 Click **OK**. The Update Run dialog box closes and QuickTest begins running the component update. The text **Update Run** flashes in the status bar while the component is being updated.

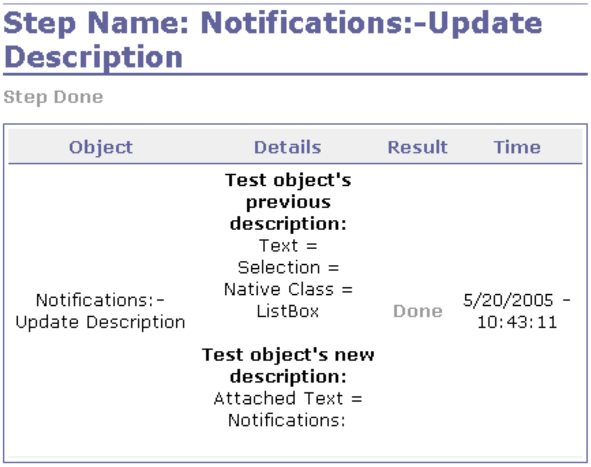
QuickTest runs the component and updates the test object descriptions. When the run session ends, the Test Results window opens.

For information on viewing the results, see Chapter 21, “Analyzing Test 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 9, “Setting Global Testing Options.”

Tip: Remember to exit the update run mode by clicking the **Update Run Mode** button when you have finished updating your component.

When the update run ends, the Test Results window can show updated test object descriptions. For example:



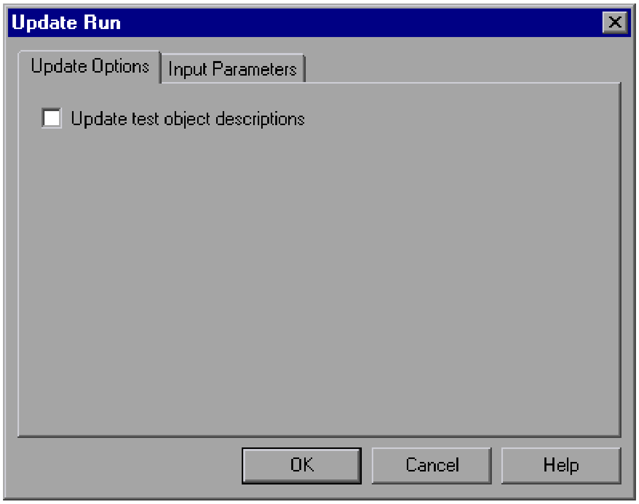
Step Name: Notifications:-Update Description

Step Done

Object	Details	Result	Time
Notifications:-Update Description	**Test object's previous description:** Text = Selection = Native Class = ListBox **Test object's new description:** Attached Text = Notifications:	Done	5/20/2005 - 10:43:11

Understanding the Update Options Tab

The Update Options tab enables you to update test object descriptions. The results of the updated component are used for subsequent runs.



Update Run

Update Options | Input Parameters

□ Update test object descriptions

The Update Options tab contains the **Update test object descriptions** check box. Selecting this option instructs QuickTest to update the test object descriptions for your business component according to the properties currently defined in the Object Identification dialog box for each object class. You can use this option to modify the set of properties used to identify an object. When you use this option, all values are updated, even if they are parameterized or use regular expressions.

Tip: 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 344.

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 component 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 also use the **Update test object descriptions** option to update the test object description property values.

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.

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.

21

Analyzing Test 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.

About Analyzing Test 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, including:

- 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
- detailed explanations of each step pass or failure, at each stage of the component

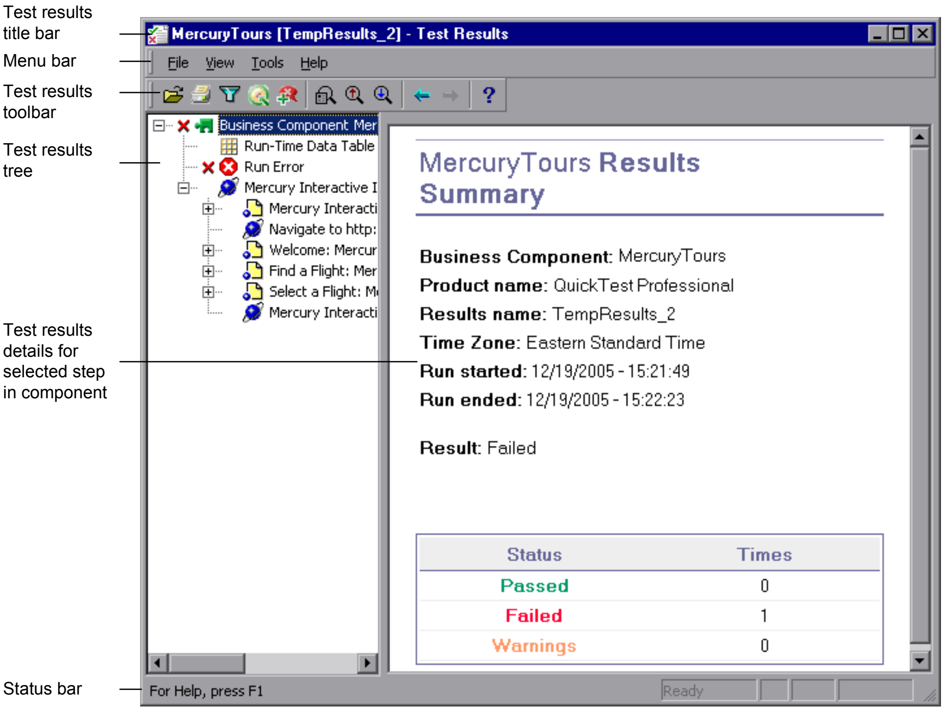
Note: The Test Results window can show results with up to 300 levels in the tree hierarchy. If you have results with more than 300 nested levels, you can view the entire report by manually opening the **results.xml** file.

Understanding 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 220.

Note: 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:



MercuryTours Results Summary

Business Component: MercuryTours
Product name: QuickTest Professional
Results name: TempResults_2
Time Zone: Eastern Standard Time
Run started: 12/19/2005 - 15:21:49
Run ended: 12/19/2005 - 15:22:23

Result: Failed

Status	Times
Passed	0
Failed	1
Warnings	0

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.
- **Test results toolbar.** Contains buttons for viewing run session results (choose **View** > **Test Results Toolbar** to display the toolbar). For more information, see "Test Results Toolbar" on page 545.
- **Test results tree.** Contains a graphic representation of the run results in the test results tree. For more information, see "Test Results Tree", below.
- **Test results details.** Contains details of the selected step. For more information, see "Test Results Details" on page 545.
- **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 546.

Test Results Tree

The left pane in the Test Results window displays the **test 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 action or test) 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.

You can collapse or expand a branch in the test results tree to change the level of detail that the tree displays.

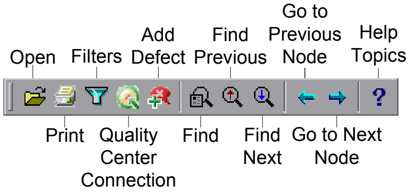
Test Results Details

By default, when the Test Results window opens, a component summary is displayed in the right pane of the window. When you select a branch or step in the tree, the right pane displays detailed information for the selected item.

The Test Results details area indicates the component name, product name (for a component), results name, the start and end date and time of the run, and whether an iteration passed or failed. For a component, the possible results are **Done** or **Failed**.

Test Results Toolbar

The Test 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 30.

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 220.)

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.

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 “Understanding the Test Results Window” on page 543.
   - 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 546.

2. You can collapse or expand a branch in the test 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 test 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 test 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.

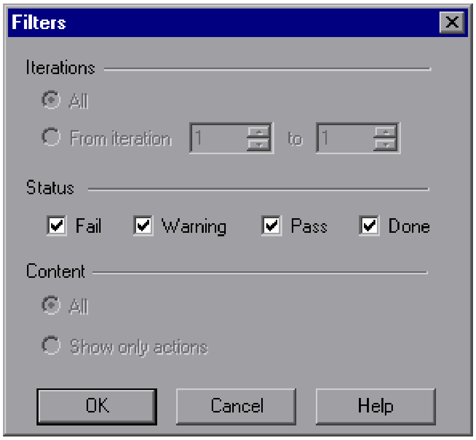
3 You can view the results of an iteration or step. The results can be one of three types:

- Steps that were not successful, but did not cause the component to stop running, are marked **Warning** in the bottom 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.



Filters

Iterations

● All

○ From iteration 1 to 1

Status

☑ Fail ☑ Warning ☑ Pass ☑ Done

Content

● All

○ Show only actions

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**.

6 To move between previously selected nodes within the test 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 550.

8 To print run results, click the **Print** button or choose **File** > **Print**. For more information, see "Printing Run Session Results" on page 553.
(You can preview the run results before you print them. For more information, see "Previewing Test Results" on page 554.)

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 571.

---

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 556.

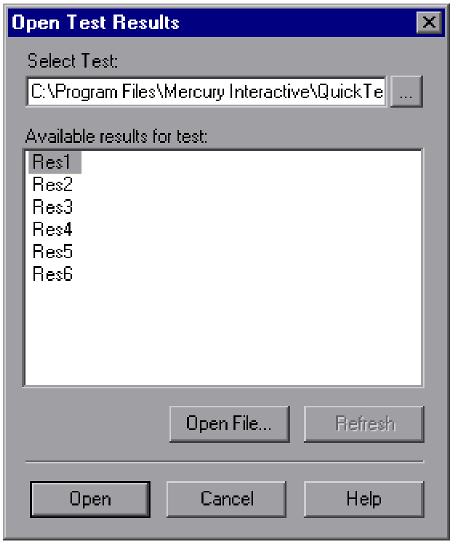
10 Choose **File** > **Exit** to close the Test Results window.

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.



Open Test Results

Select Test: C:\Program Files\Mercury Interactive\QuickTe

Available results for test:

Res1
Res2
Res3
Res4
Res5
Res6

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 530.

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.

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. These run results contain the same information described in "Understanding the Test Results Window" on page 543 plus the following additional fields:

Test set. Specifies the location of the business process test.

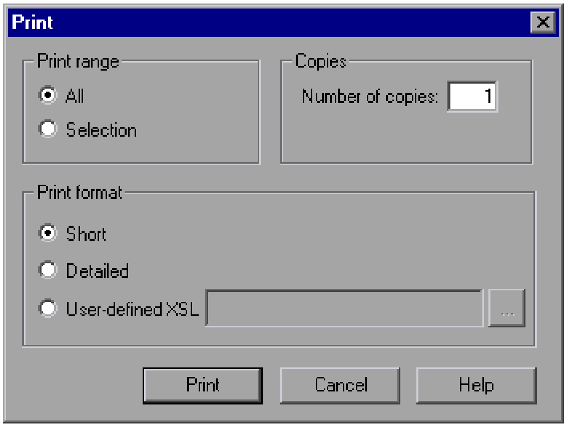
Test instance. Specifies the instance number of the test in the business process test. For example, if the same test is included twice in the business process test, you can view the results of Test instance 1 and Test instance 2.

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.



Print

Print range
● All
○ Selection

Copies
Number of copies: 1

Print format
● Short
○ Detailed
○ User-defined XSL

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 test 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 test results tree. This option is only available if you selected **All** in step 2.
- **Detailed.** Prints all available information for each item in the test 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 572.

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.

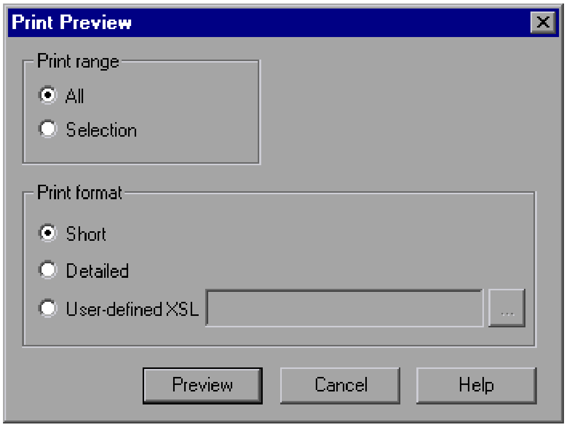
Previewing Test Results

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.



Print Preview

Print range
● All
○ Selection

Print format
● Short
○ Detailed
○ User-defined XSL

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 test results tree.

3 Select a **Print format** option:

- **Short.** Previews a summary line (when available) for each item in the test results tree. This option is only available if you selected **All** in step 2.
- **Detailed.** Previews all available information for each item in the test 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 572.

4 Click **Preview** to preview the appearance of your run results on screen.

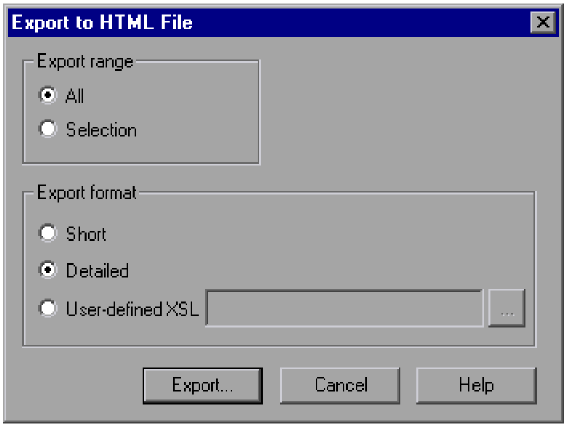
Tip: If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the **Page Setup** button in the Print Preview window and change the page orientation from **Portrait** to **Landscape**.

Exporting Test Results

You can export the run results from the Test Results window to an HTML file. This enables you to easily view the run results when you are not in a QuickTest environment. For example, you can send the HTML 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.



Export to HTML File

Export range

● All

○ Selection

Export format

○ Short

● Detailed

○ User-defined XSL

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 test results tree.

3 Select an **Export format** option:

- ➤ **Short.** Exports a summary line (when available) for each item in the test results tree. This option is only available if you selected **All** in step 2.
- ➤ **Detailed.** Exports all available information for each item in the test 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 572.

Note: The **Export format** options are available only for run results created with QuickTest version 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.

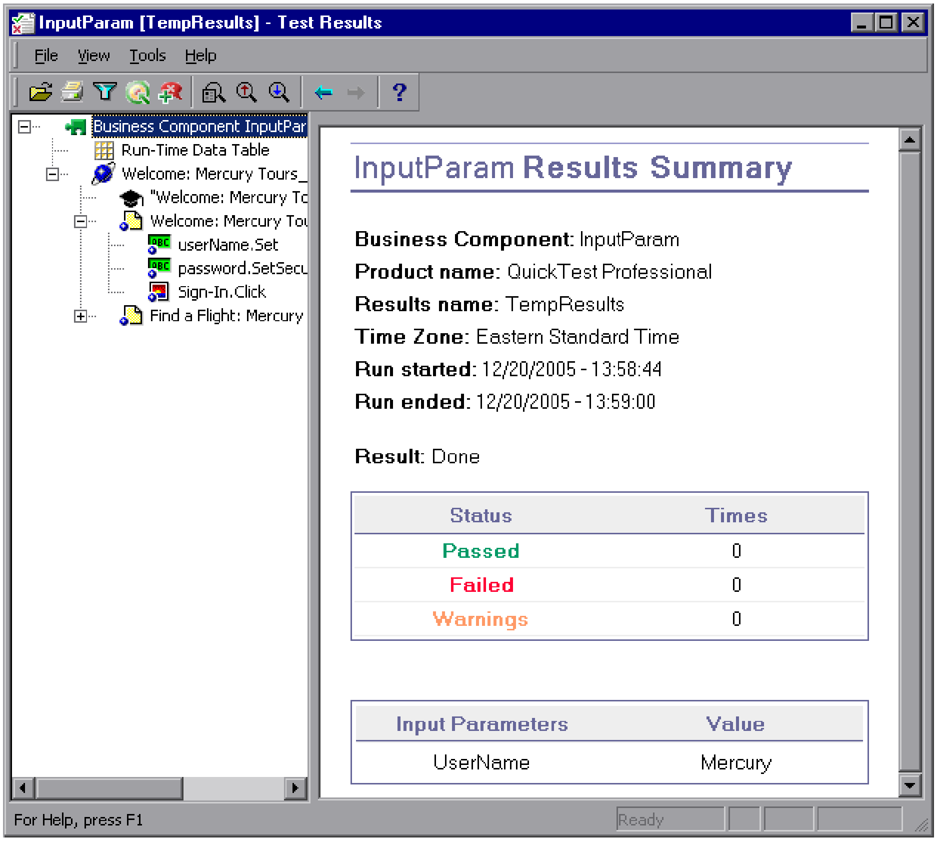
Viewing Parameterized Values in the Test Results Window

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 546.
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.



InputParam Results Summary

Business Component: InputParam
Product name: QuickTest Professional
Results name: TempResults
Time Zone: Eastern Standard Time
Run started: 12/20/2005 - 13:58:44
Run ended: 12/20/2005 - 13:59:00

Result: Done

Status	Times
Passed	0
Failed	0
Warnings	0

Input Parameters	Value
UserName	Mercury

The example above shows the input parameter UserName defined for the component with the value Mercury.

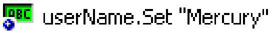
For more information on defining and using parameters in your components, see "Working with Parameters" on page 170.

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.

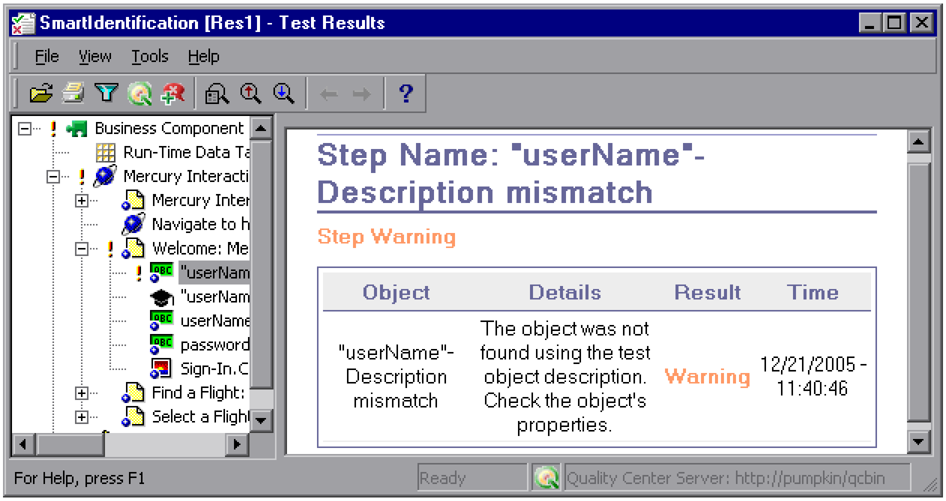
Smart Identification—No Object Matches the Recorded Description

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: "userName"- Description mismatch	An indication that the object (for example, the userName WebEdit object) was not found.
A Smart Identification icon for the missing object. For example: "userName"- Smart Identification	An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to modify the recorded test object description, so that QuickTest can find the object using the description in future run sessions.
The actual step performed. For example:  userName.Set "Mercury"	Normal result details for the performed step.

For more information on the Smart Identification mechanism, see Chapter 15, “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.



Step Name: "userName"- Description mismatch

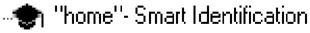
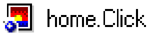
Step Warning

Object	Details	Result	Time
"userName"- Description mismatch	The object was not found using the test object description. Check the object's properties.	Warning	12/21/2005 - 11:40:46

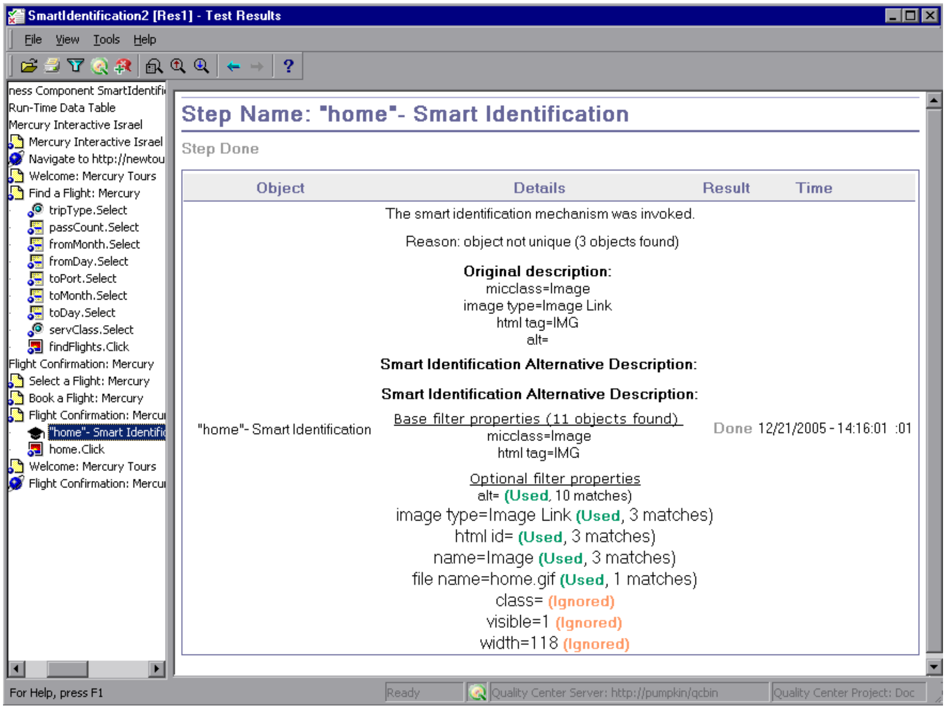
Smart Identification—Multiple Object Match the Recorded Description

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:  "home"- Smart Identification	An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to create a unique object description for the object, so that QuickTest can find the object using the description in future run sessions.
The actual step performed. For example:  home.Click	Normal result details for the performed step.

The image below shows the results for a component in which Smart Identification was used to uniquely identify the Home object after the recorded description resulted in multiple matches.


SmartIdentification2 [Res1] - Test Results
File
View
Tools
Help
For Help, press F1
Ready
Quality Center Server: http://pumpkin/qcbin
Quality Center Project: Doc

ness Component SmartIdentifi
Run-Time Data Table
Mercury Interactive Israel
Mercury Interactive Israel
Navigate to http://newtou
Welcome: Mercury Tours
Find a Flight: Mercury
tripType.Select
passCount.Select
fromMonth.Select
fromDay.Select
toPort.Select
toMonth.Select
toDay.Select
servClass.Select
findFlights.Click
Flight Confirmation: Mercury
Select a Flight: Mercury
Book a Flight: Mercury
Flight Confirmation: Mercu
"home"- Smart Identifi
home.Click
Welcome: Mercury Tours
Flight Confirmation: Mercu

Step Name: "home"- Smart Identification

Step Done

Object	Details	Result	Time
"home"- Smart Identification	The smart identification mechanism was invoked. Reason: object not unique (3 objects found) **Original description:** micclass=Image image type=Image Link html tag=IMG alt= **Smart Identification Alternative Description:** **Smart Identification Alternative Description:** Base filter properties (11 objects found) micclass=Image html tag=IMG Optional filter properties alt= **(Used**, 10 matches) image type=Image Link **(Used**, 3 matches) html id= **(Used**, 3 matches) name=Image **(Used**, 3 matches) file name=home.gif **(Used**, 1 matches) class= **(Ignored)** visible=1 **(Ignored)** width=118 **(Ignored)**	Done	12/21/2005 - 14:16:01 :01

If the Smart Identification mechanism cannot successfully identify the object, the component fails and a normal failed step is displayed in the Test Results.

Deleting Test 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.

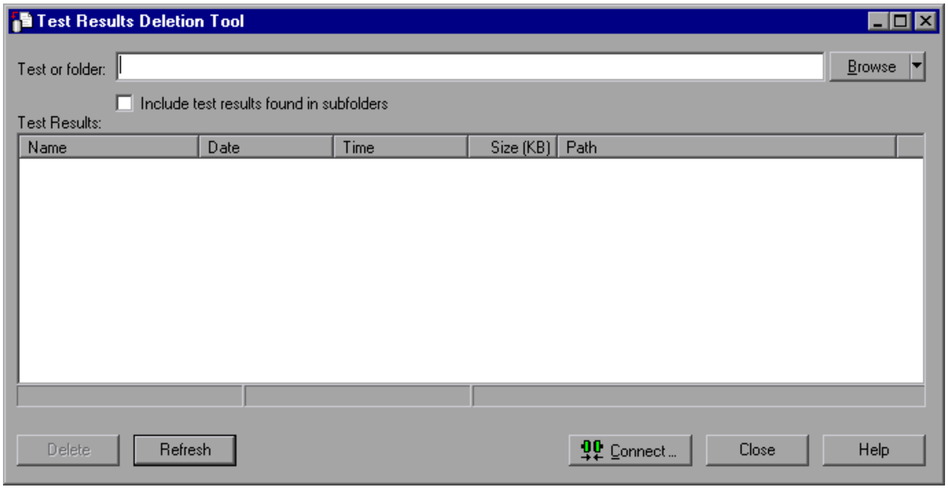
Deleting Results Using the Test Results Deletion Tool

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 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 22.

---

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. For more information, see "Command Line Options," below.

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 test you specified. The log file indicates which results were deleted and the reasons why other results were not. For example, results may not be deleted if they are smaller than the minimum file size you specified.

You can specify a file path and name or use the default path and name. If you do not specify a file name, the default log file name is **TestResultsDeletionTool.log** in the folder where the 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 tests in Quality Center.

The following example deletes all results larger than or equal to 10000 bytes. Results that are smaller than 10000 bytes are not deleted.

```
TestResultsDeletionTool.exe  -Silent -Test "C:\tests\test1" -MinSize "10000"
```

-Name *result_file_name*

Specifies the name(s) of the result file(s) to be deleted. Only results with the specified name(s) are deleted.

You can use regular expressions to specify criteria for the result file(s) you want to delete. For more information on regular expressions and regular expression syntax, refer to the *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 folder and its subfolders. When using the **-Recursive** option, the **-Test** option should contain the path of the folder that contains the tests results you want to delete (and not the path of a specific test).

The following example deletes all results in the **F:\Tests** folder and all of its subfolders.

```
TestResultsDeletionTool.exe  -Test "F:\Tests" -Recursive
```

Note: The **-Recursive** option is available only for folders in the file system. It is not supported when working with tests 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 *test_or_folder_path*

Sets the test or test path from which the Test Results Deletion Tool deletes test results. You can specify a test name and path, file system path, or full Quality Center path.

This option is available only when used in conjunction with the **-Silent** option.

Note: The **-Domain**, **-Server**, **-Project**, **-User**, and **-Password** options must be used to connect to Quality Center.

The following example opens the 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"
```

Note: The **-Test** option can be combined with the **-Recursive** option to delete all test results in the specified 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, refer to the *QuickTest Professional User's Guide*. For more information on Quality Center, refer to the *Mercury Quality Center User's Guide*.

To manually submit a defect to Quality Center:

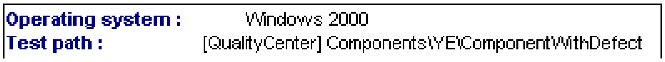
1 Choose **Tools** > **Quality Center Connection** or click the **Quality Center Connection** button to connect to a Quality Center project. For more information on connecting to Quality Center, see "Connecting to Your Quality Center Project" on page 22.

Note: If you do not connect to a Quality Center project before proceeding to the next step, QuickTest prompts you to connect before continuing.

---

2 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.

3 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
```

4 Click **Submit** to add the defect information to the Quality Center project.

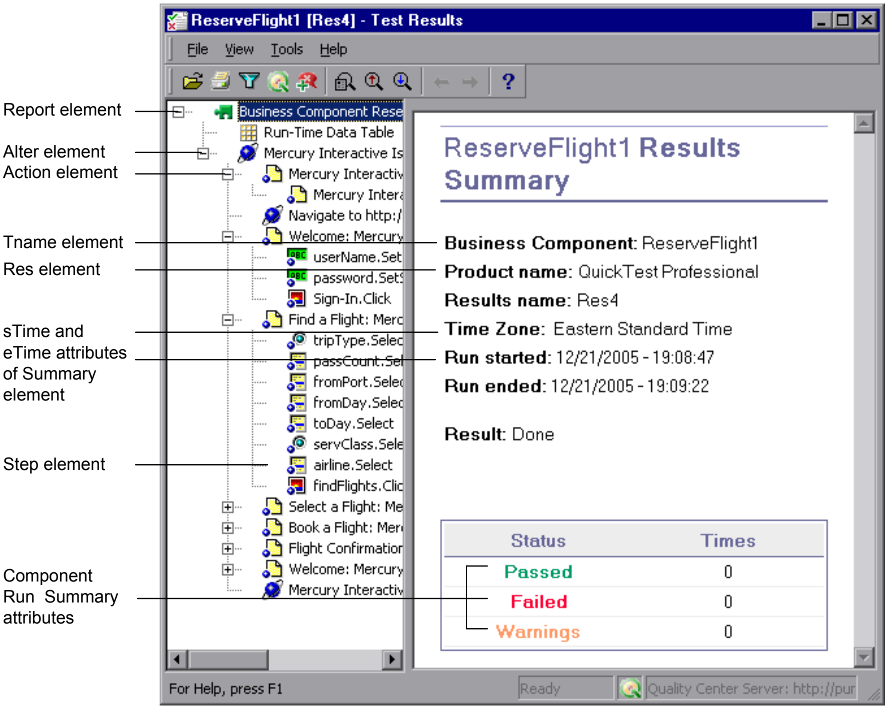
5 Click **Close** to close the Add Defect dialog box.

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 test 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.



ReserveFlight1 Results Summary

Business Component: ReserveFlight1
Product name: QuickTest Professional
Results name: Res4
Time Zone: Eastern Standard Time
Run started: 12/21/2005 - 19:08:47
Run ended: 12/21/2005 - 19:09:22

Result: Done

Status	Times
Passed	0
Failed	0
Warnings	0

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 546.

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 553.

For more information on exporting the test results to an HTML file using a customized **.xsl** file, see “Exporting Test Results” on page 556.

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, refer to the XML Report Help (**Help** > **QuickTest Professional Help** > **QuickTest Advanced References** > **QuickTest Test Results Schema**).

Part VI

Working with Advanced Features

22

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.

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 580.

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 operation(s) 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 Operation(s).** The operation(s) 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 104.

When you run a component for which you have defined recovery scenarios and an error occurs, QuickTest looks for the defined trigger event(s) 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 619.

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.

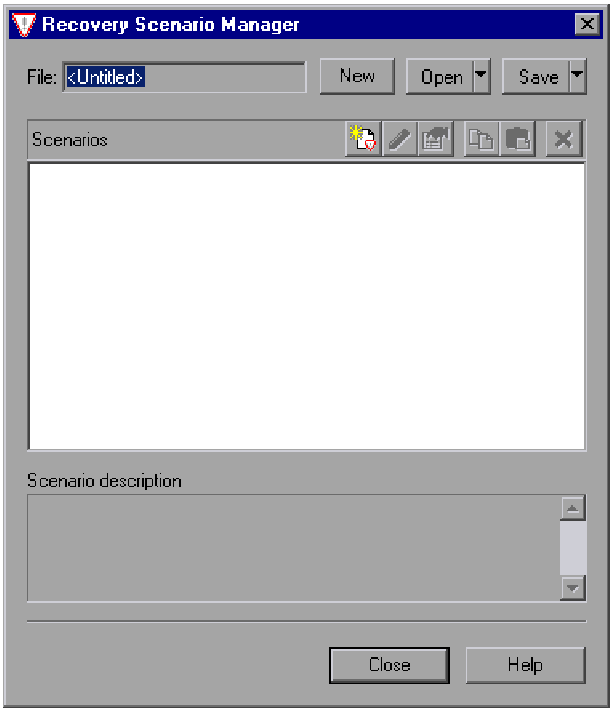
Using the Recovery Scenario Manager dialog box, you can then select any recovery file to manage all of the recovery scenarios stored in that file. This enables you to edit a selected recovery scenario, associate specific recovery scenarios with specific components to instruct QuickTest to implement the 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.



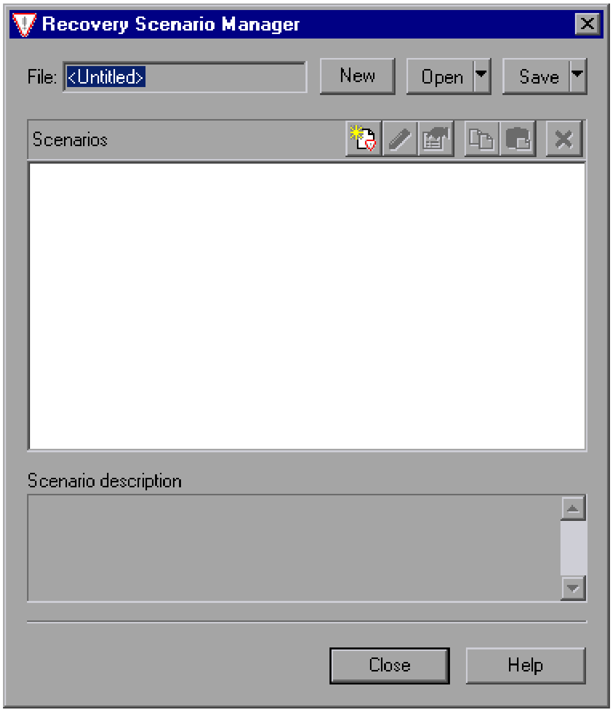
2 By default, the Recovery Scenario Manager dialog box opens with a new recovery file. You can either use this new file, or click the **Open** button to choose an existing recovery file. Alternatively, you can click the arrow next to the **Open** button to select a recently-used recovery file from the 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 scenario(s) saved in the recovery file, and a description of each scenario.



The Recovery Scenario Manager dialog box contains the following toolbar buttons:

Option	Description
New	Creates a new recovery file. For more information, see “Creating a Recovery File” on page 581.
Open ▾	Opens an existing recovery file. You can also click the arrow to select a recovery file from the list of recently-used recovery files.
Save ▾	Saves the current recovery file. For more information, see “Saving the Recovery Scenario in a Recovery File” on page 610.
✳	Opens the Recovery Scenario Wizard, in which you define a new recovery scenario. For more information, see “Understanding the Recovery Scenario Wizard” on page 585.
✎	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 613.
🗒	Displays summary properties for the selected recovery scenario in read-only format. For more information, see “Viewing Recovery Scenario Properties” on page 612.
⧉	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 614.
📋	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 614.
✕	Deletes a recovery scenario. For more information, see “Deleting Recovery Scenarios” on page 614.

Note: Each recovery scenario is represented by an icon that indicates its type. For more information, see “Managing Recovery Scenarios” on page 611.

Understanding the Recovery Scenario Wizard

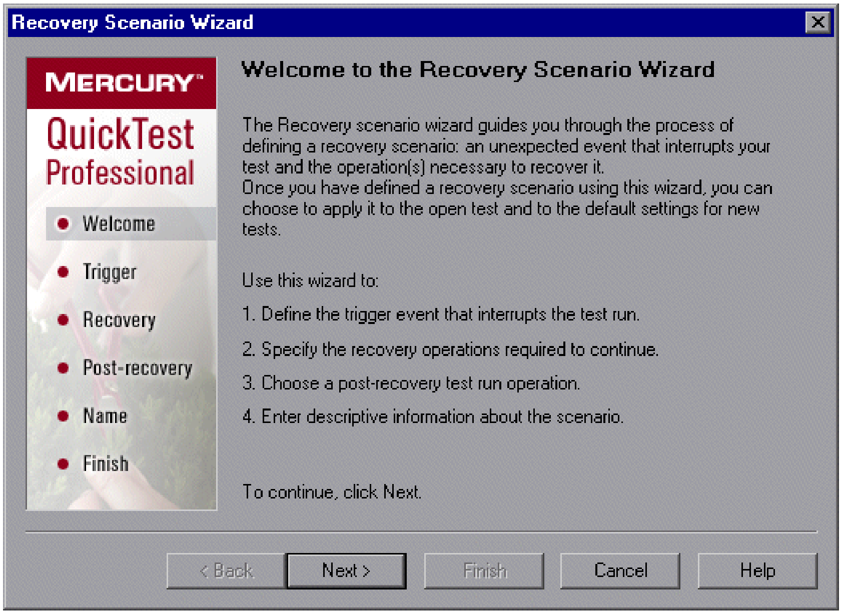
The Recovery Scenario Wizard leads you, step-by-step, through the process of creating a recovery scenario. The Recovery Scenario Wizard contains the following main steps:

- defining the trigger event that interrupts the run session
- specifying the recovery operation(s) 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.



Welcome to the Recovery Scenario Wizard

The Recovery scenario wizard guides you through the process of defining a recovery scenario: an unexpected event that interrupts your test and the operation(s) necessary to recover it.
Once you have defined a recovery scenario using this wizard, you can choose to apply it to the open test and to the default settings for new tests.

Use this wizard to:

1. Define the trigger event that interrupts the test run.

2. Specify the recovery operations required to continue.

3. Choose a post-recovery test run operation.

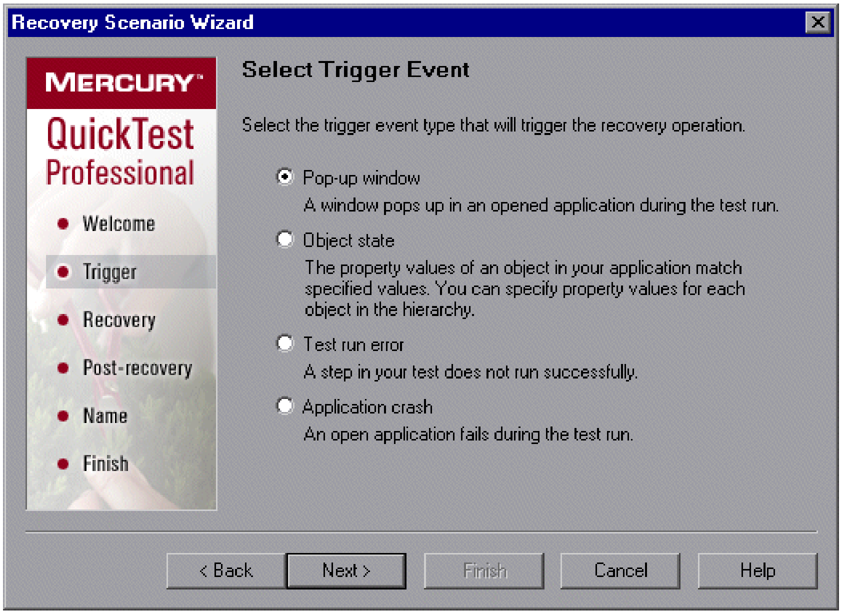
4. Enter descriptive information about the scenario.

To continue, click Next.

Click **Next** to continue to the Select Trigger Event Screen (described on page 586).

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 Trigger Event

Select the trigger event type that will trigger the recovery operation.

● Pop-up window
A window pops up in an opened application during the test run.

○ Object state
The property values of an object in your application match specified values. You can specify property values for each object in the hierarchy.

○ Test run error
A step in your test does not run successfully.

○ Application crash
An open application fails during the test run.

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 589).

- **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 591).

- **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 594).

- **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 597).

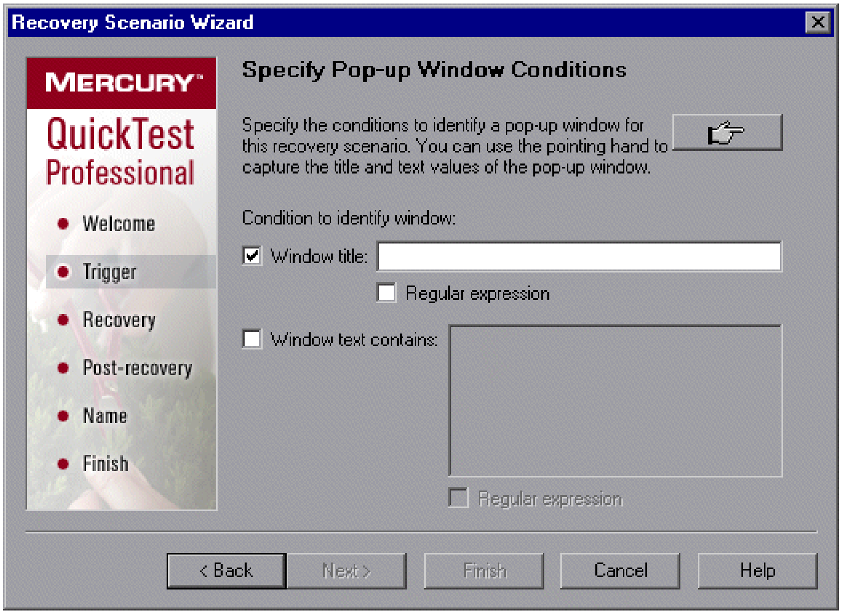
Notes:

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 586), the Specify Pop-up Window Conditions screen opens.



Recovery Scenario Wizard

Specify Pop-up Window Conditions

Specify the conditions to identify a pop-up window for this recovery scenario. You can use the pointing hand to capture the title and text values of the pop-up window.

Condition to identify window:

☑ Window title:

□ Regular expression

□ Window text contains:

□ Regular expression

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, refer to the *QuickTest Professional User's Guide*.
- Click the pointing hand and then click the pop-up window to capture the window title and textual content of the window.

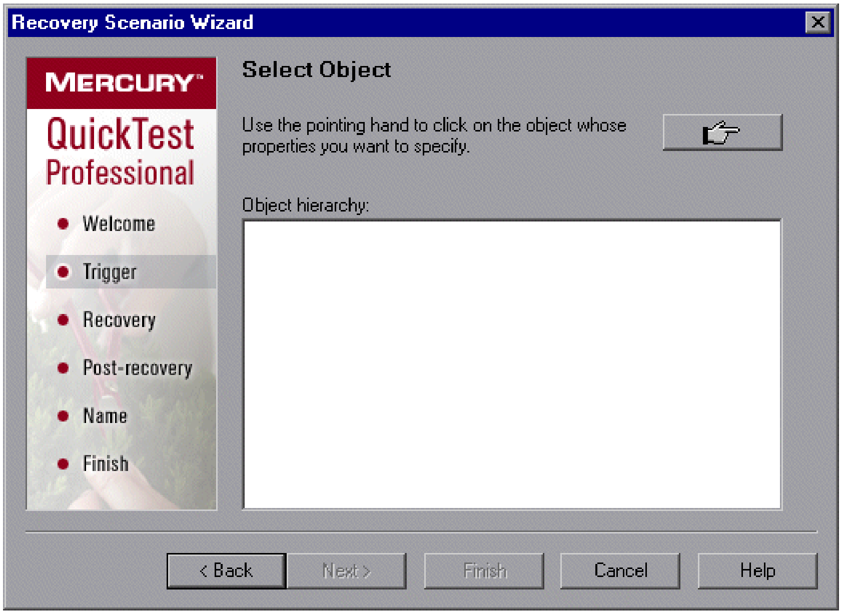
Note: Using the first option (**Window title** and/or **Window text**) instructs QuickTest to identify any pop-up window that contains the relevant title and/or text. Using the second option (pointing hand) instructs QuickTest to identify only pop-up windows that match the object property values of the 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 597).

Select Object Screen

If you chose an **Object state** trigger in the Select Trigger Event Screen (described on page 586), the Select Object screen opens.



Select Object

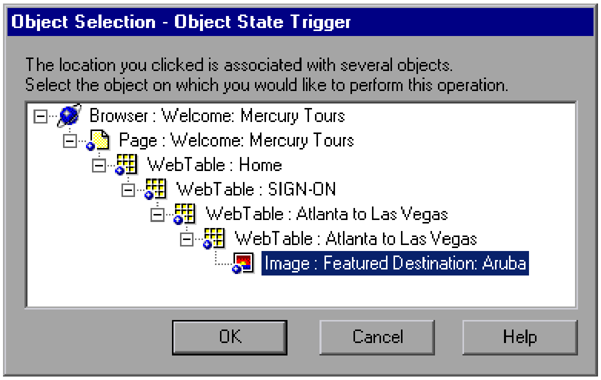
Use the pointing hand to click on the object whose properties you want to specify.

Object hierarchy:

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.



Object Selection - Object State Trigger

The location you clicked is associated with several objects.
Select the object on which you would like to perform this operation.

- Browser : Welcome: Mercury Tours
  - Page : Welcome: Mercury Tours
    - WebTable : Home
      - WebTable : SIGN-ON
        - WebTable : Atlanta to Las Vegas
          - WebTable : Atlanta to Las Vegas
            - Image : Featured Destination: Aruba

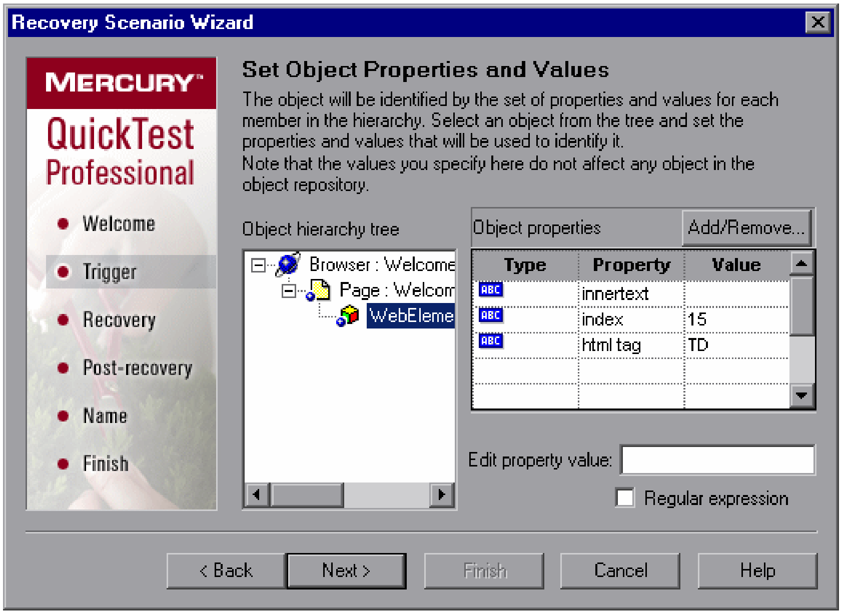
Select the object whose properties you want to specify and click **OK**. The selected object and its parents are displayed in the Select Object screen.

Note: The hierarchical object selection tree also enables you to select an object that QuickTest would not ordinarily record (a non-parent object), such as a web table.

Click **Next** to continue to the Set Object Properties and Values Screen (described on page 593).

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 591), the Set Object Properties and Values screen opens.



Set Object Properties and Values

The object will be identified by the set of properties and values for each member in the hierarchy. Select an object from the tree and set the properties and values that will be used to identify it.
Note that the values you specify here do not affect any object in the object repository.

Type	Property	Value
ABC	innertext	
ABC	index	15
ABC	html tag	TD

Edit property value:

□ Regular expression

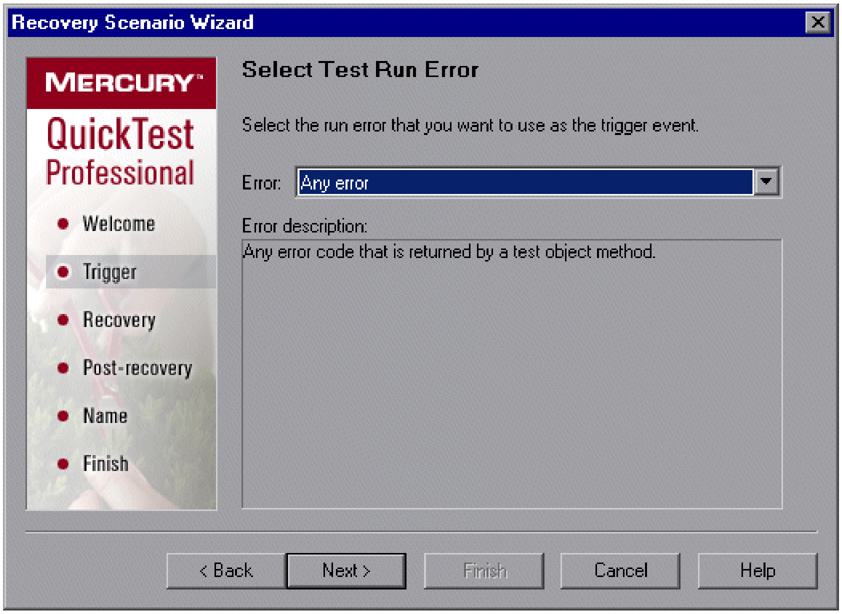
For each object in the hierarchy, in the **Edit property value** box, you can modify the property values used to identify the object. You can also click the **Add/Remove** button to add or remove object properties from the list of property values to check. Note that an object is identified only by its 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, refer to the *QuickTest Professional User's Guide*.

Click **Next** to continue to the Recovery Operations Screen (described on page 597).

Select Test Run Error Screen

If you chose a **Test run error** trigger in the Select Trigger Event Screen (described on page 586), the Select Test Run Error screen opens.



Select Test Run Error

Select the run error that you want to use as the trigger event.

Error: Any error

Error description:
Any error code that is returned by a test object method.

In the **Error** list, choose the run error that you want to use as the trigger event:

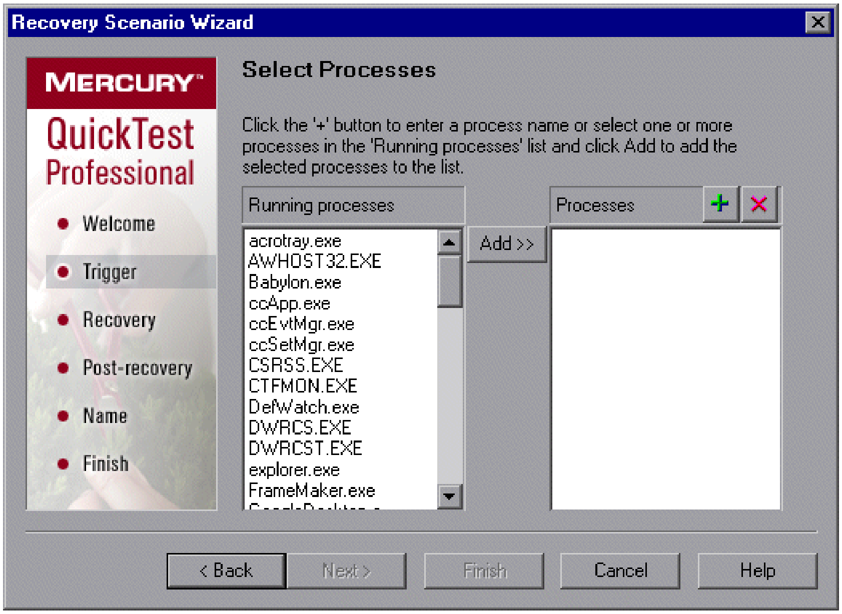
- **Any error.** Any error code that is returned by a test object method.
- **Item in list or menu is not unique.** Occurs when more than one item in the list, menu, or tree has the name specified in the method argument.
- **Item in list or menu not found.** Occurs when QuickTest cannot identify the 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 597.

Select Processes Screen

If you chose an **Application crash** trigger in the Select Trigger Event Screen (described on page 586), the Select Processes screen opens.



Select Processes

Click the '+' button to enter a process name or select one or more processes in the 'Running processes' list and click Add to add the selected processes to the list.

Running processes

acrotray.exe
AWHOST32.EXE
Babylon.exe
ccApp.exe
ccEvtMgr.exe
ccSetMgr.exe
CSRSS.EXE
CTFMON.EXE
DefWatch.exe
DWRCS.EXE
DWRCST.EXE
explorer.exe
FrameMaker.exe

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.

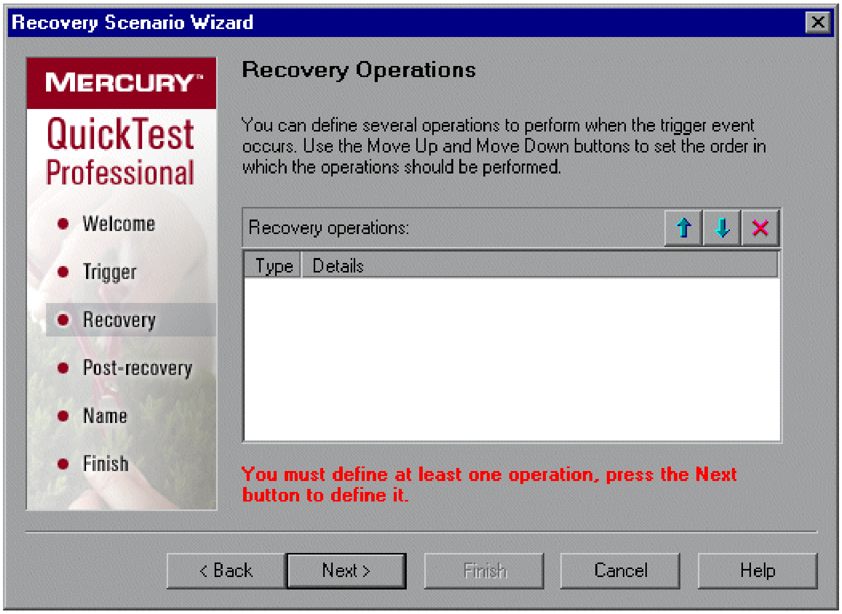
- ➤ To add a process from the **Running processes** list, double-click a process in the **Running processes** list or select it and click the **Add** button. You can select multiple processes using standard Windows multiple selection techniques (CTRL and SHIFT keys).
- ➤ To add a process directly to 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 597).

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.



Recovery Scenario Wizard

Recovery Operations

You can define several operations to perform when the trigger event occurs. Use the Move Up and Move Down buttons to set the order in which the operations should be performed.

Recovery operations:

Type	Details

You must define at least one operation, press the Next button to define it.

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 598).

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.

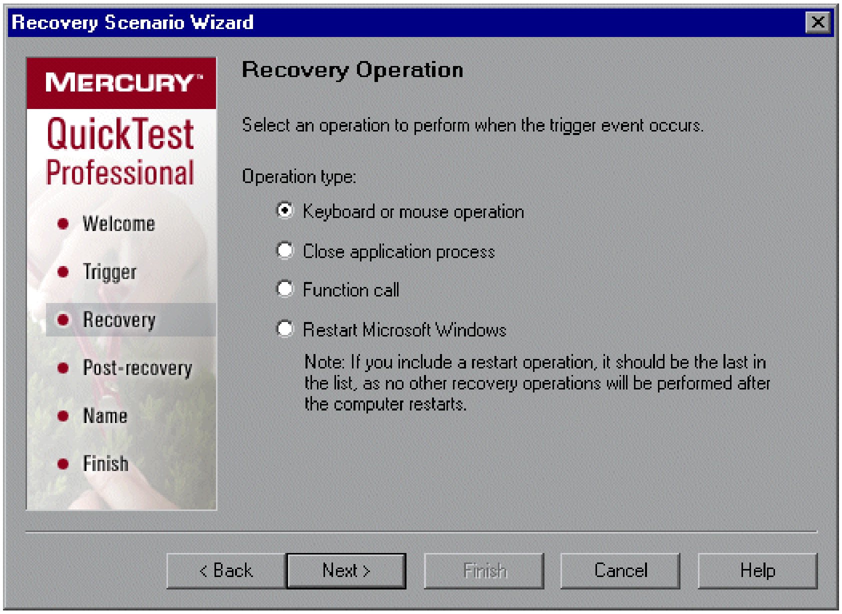
Note: If you define a **Restart Microsoft Windows** recovery operation, it is always inserted as the last recovery operation, and you cannot change its position in the list.

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 606).

Recovery Operation Screen

The Recovery Operation screen enables you to specify the operation(s) QuickTest performs after it detects the trigger event.



Recovery Scenario Wizard

Recovery Operation

Select an operation to perform when the trigger event occurs.

Operation type:

● Keyboard or mouse operation

○ Close application process

○ Function call

○ Restart Microsoft Windows

Note: If you include a restart operation, it should be the last in the list, as no other recovery operations will be performed after the computer restarts.

Select a type of recovery operation and click **Next**. The next screen displayed in the wizard depends on which recovery operation type you select.

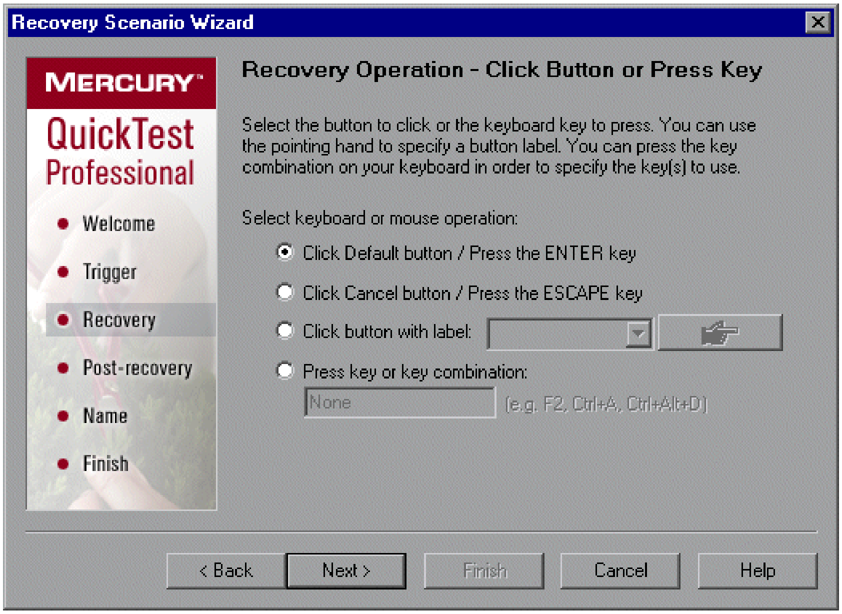
You can define the following types of recovery operations:

- **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 600).
- **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 602).
- **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 603).
- **Restart Microsoft Windows.** QuickTest restarts Microsoft Windows. Select this option and click **Next** to continue to the Recovery Operations Screen (described on page 597).

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 598), the Recovery Operation – Click Button or Press Key screen opens.



Recovery Scenario Wizard

Recovery Operation - Click Button or Press Key

Select the button to click or the keyboard key to press. You can use the pointing hand to specify a button label. You can press the key combination on your keyboard in order to specify the key(s) to use.

Select keyboard or mouse operation:

● Click Default button / Press the ENTER key

○ Click Cancel button / Press the ESCAPE key

○ Click button with label:

○ Press key or key combination:

None (e.g. F2, Ctrl+A, Ctrl+Alt+D)

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.

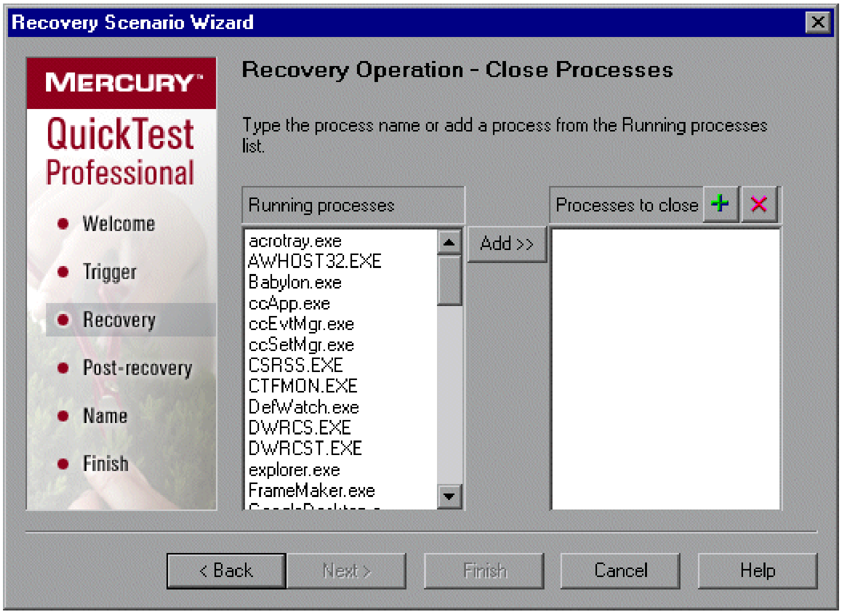
  All button labels in the selected window are displayed in the list box. Select the required button from the list.

- **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 598), the Recovery Operation – Close Processes screen opens.



Recovery Scenario Wizard

Recovery Operation - Close Processes

Type the process name or add a process from the Running processes list.

Running processes

acrotray.exe
AWHOST32.EXE
Babylon.exe
ccApp.exe
ccEvtMgr.exe
ccSetMgr.exe
CSRSS.EXE
CTFMON.EXE
DefWatch.exe
DWRCS.EXE
DWRCST.EXE
explorer.exe
FrameMaker.exe

Processes to close

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.

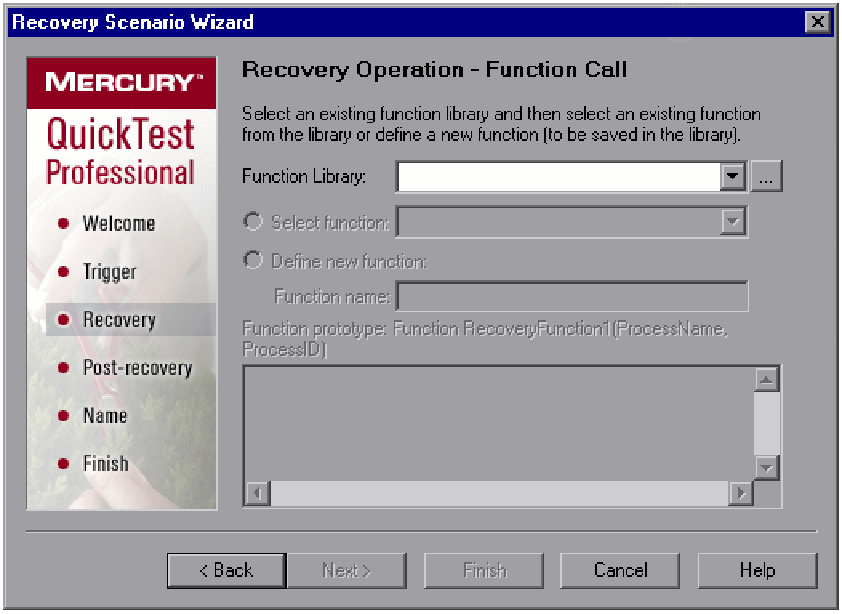
➤ To remove a process from the **Processes to close** list, select it and click the **Remove Process** button.

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 598), the Recovery Operation – Function Call screen opens.



Recovery Scenario Wizard

Recovery Operation - Function Call

Select an existing function library and then select an existing function from the library or define a new function (to be saved in the library).

Function Library:

○ Select function:

○ Define new function:

Function name:

Function prototype: Function RecoveryFunction1(ProcessName, ProcessID)

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.

Note: Only functions that match the prototype syntax for the trigger type selected in the "Select Trigger Event Screen" on page 586 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.
)
```

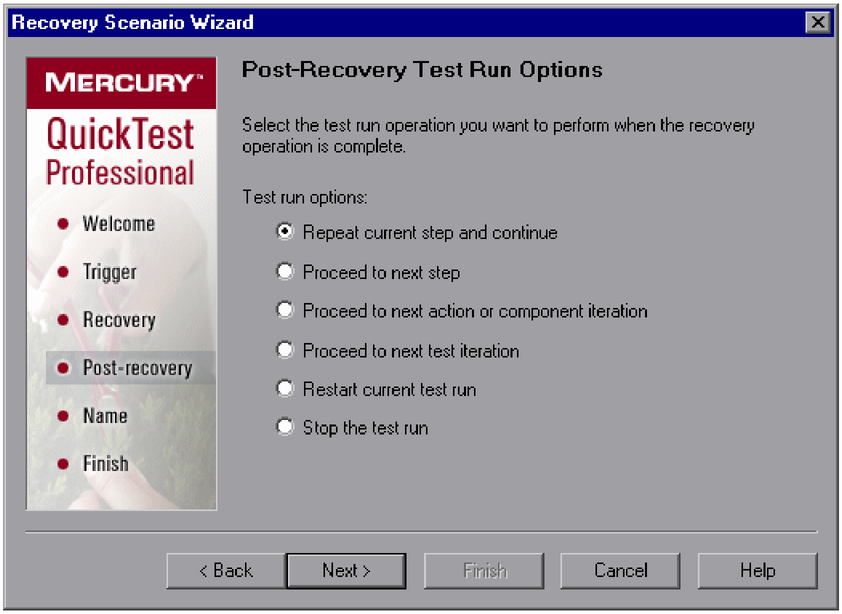
➤ **Define new function.** Create a new function by specifying a unique name for it, and defining the function in the **Function Name** box according to the displayed function prototype. The new function is added to the function library you selected.

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 597) 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 597) 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.



Post-Recovery Test Run Options

Select the test run operation you want to perform when the recovery operation is complete.

Test run options:

- ● Repeat current step and continue
- ○ Proceed to next step
- ○ Proceed to next action or component iteration
- ○ Proceed to next test iteration
- ○ Restart current test run
- ○ Stop the test run

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 617.

- **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.

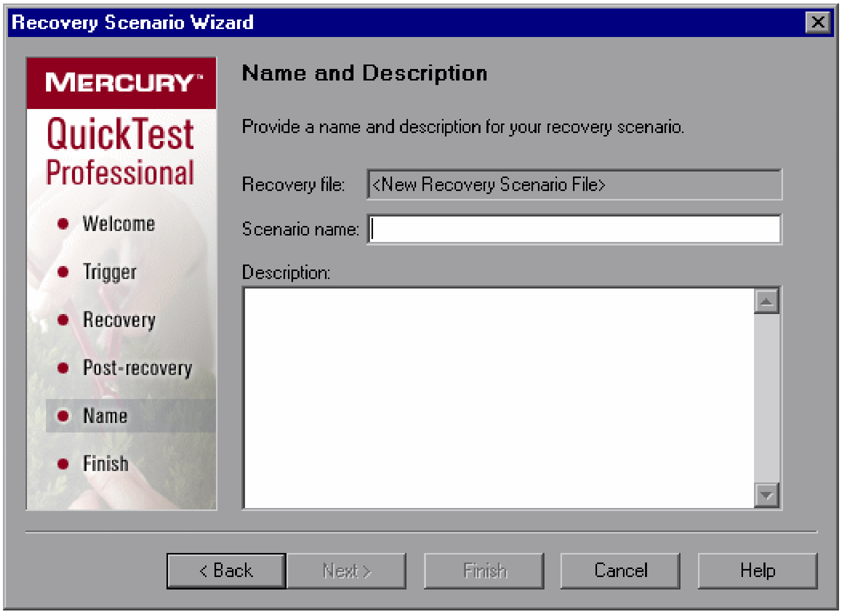
Note: If you chose **Restart Microsoft Windows** as a recovery operation, you can choose from only the last two test run options listed above.

Select a test run option and click **Next** to continue to the Name and Description Screen (described on page 608).

Name and Description Screen

After you specify a test run option in the Post-Recovery Test Run Options Screen (described on page 606), 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.



Name and Description

Provide a name and description for your recovery scenario.

Recovery file: <New Recovery Scenario File>

Scenario name:

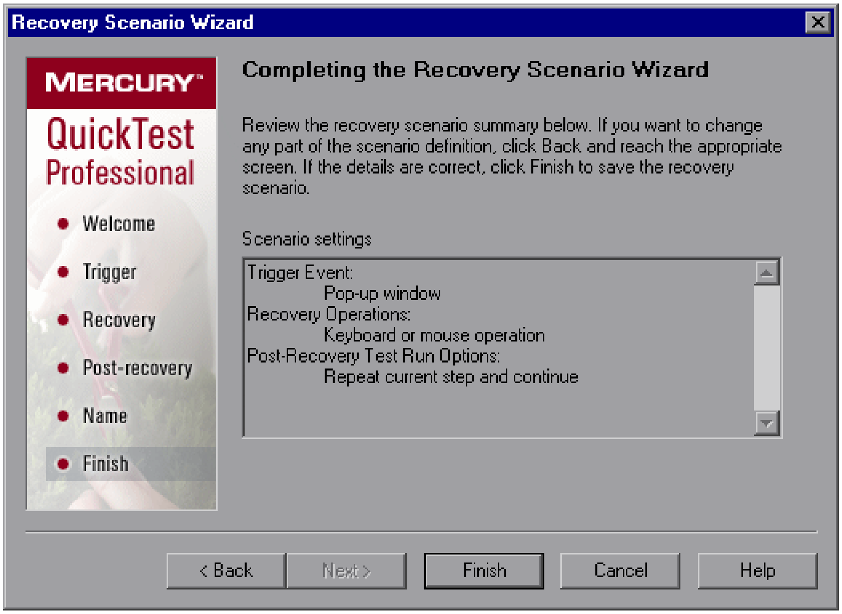
Description:

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 609).

Completing the Recovery Scenario Wizard Screen

After you specify a recovery scenario name and description in the Name and Description Screen (described on page 608) 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.



Completing the Recovery Scenario Wizard

Review the recovery scenario summary below. If you want to change any part of the scenario definition, click Back and reach the appropriate screen. If the details are correct, click Finish to save the recovery scenario.

Scenario settings

```
Trigger Event:
        Pop-up window
Recovery Operations:
        Keyboard or mouse operation
Post-Recovery Test Run Options:
        Repeat current step and continue
```

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 69.

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.

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.

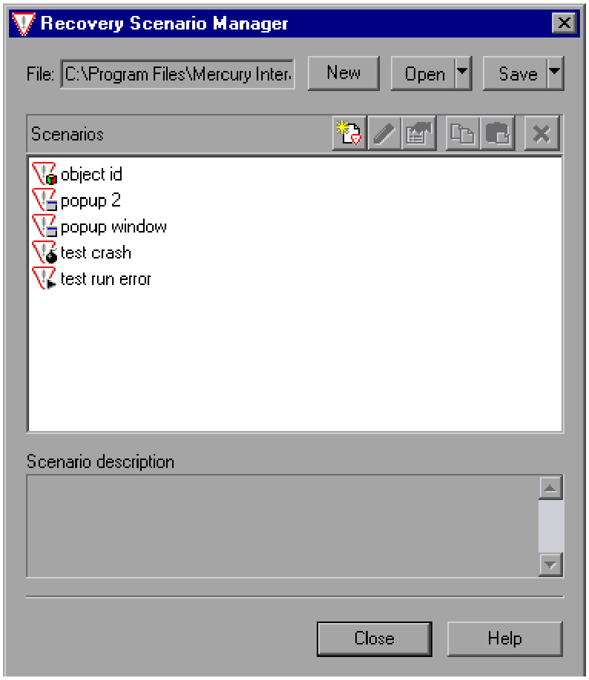
2 Choose the folder in which you want to save the file.

3 Type a name for the file in the **File name** box. The recovery file is saved in the specified location with the file extension **.qrs**.

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 above. 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.

Managing Recovery Scenarios

Once you have created recovery scenarios, you can use the Recovery Scenario Manager to manage them.



File: C:\Program Files\Mercury Inter.

- object id
- popup 2
- popup window
- test crash
- test run error

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

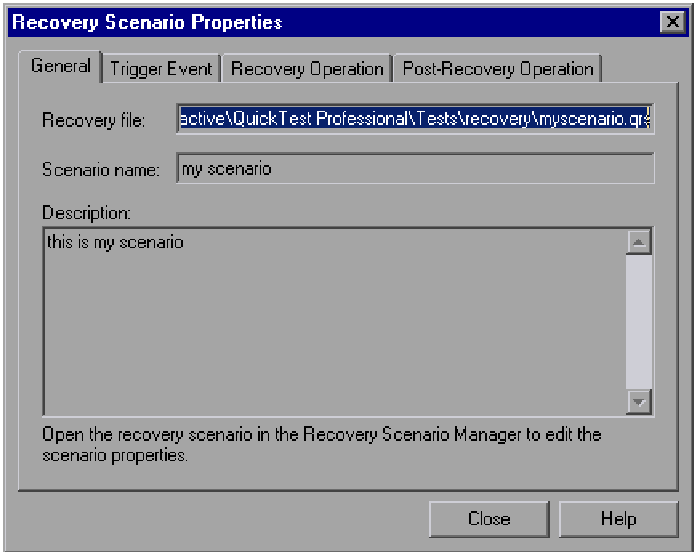
Viewing Recovery Scenario Properties

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.



Recovery Scenario Properties

Recovery file: active\QuickTest Professional\Tests\recovery\myscenario.qrs

Scenario name: my scenario

Description: this is my scenario

Open the recovery scenario in the Recovery Scenario Manager to edit the scenario properties.

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 operation(s) 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 581.

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.

Setting the Recovery Scenarios List for Your Application Areas

After you have created recovery scenarios, you associate them with selected tests or components so that QuickTest will perform the appropriate scenario(s) 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 69.

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 613.

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 612.

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 619.

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 69.

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, refer to the *QuickTest Professional Object Model Reference*.

23

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.

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 637 and “Registering User-Defined Functions as Test Object Methods” on page 652.

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, refer to 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 678.

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 635. 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 637.

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, refer to the *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 271.

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 83.

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 630.

To open an existing function library:

Perform one of the following:

➤ Choose **File** > **Open** > **Function Library**

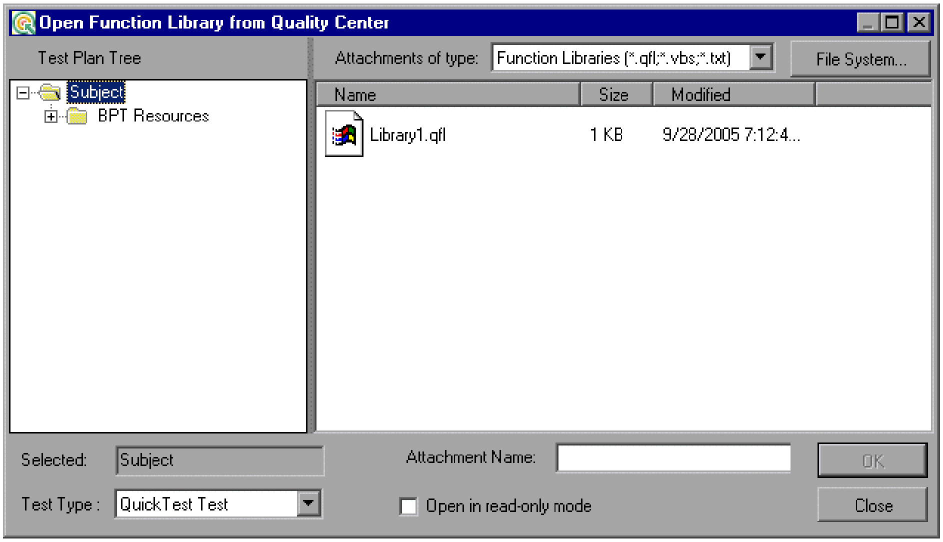
➤ Click the **Open** button down arrow and choose **Function Library**

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.



Attachments of type: Function Libraries (*.qfl;*.vbs;*.txt)

Name	Size	Modified
Library1.qfl	1 KB	9/28/2005 7:12:4...

Selected: Subject

Attachment Name:

Test Type : QuickTest Test

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 630 and "Debugging a Function Library" on page 632.

Saving a Function Library

After you create or edit a function library in QuickTest, you can save it to your Quality Center project.

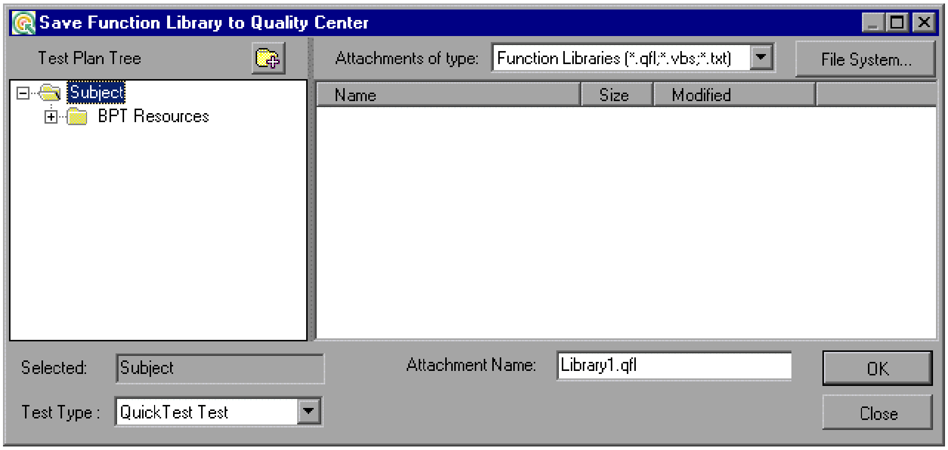
Tips:

- 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.



Save Function Library to Quality Center

Attachments of type: Function Libraries (*.qfl;*.vbs;*.txt)

Selected: Subject

Test Type : QuickTest Test

Attachment Name: Library1.qfl

3 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).

---

4 In the **Attachment name** box, type a name for the function library.

5 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.

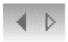
Navigating Between Open QuickTest Documents

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 673.

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 624.

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 507.

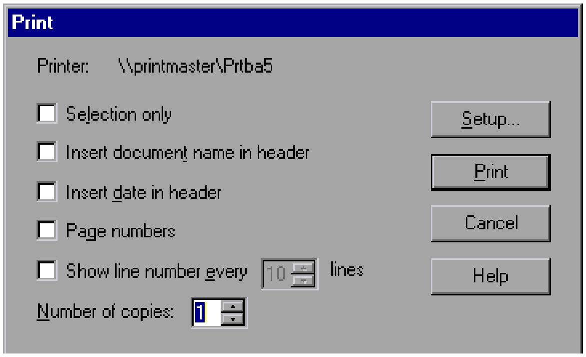
Note: During a debug session, all documents are read-only and cannot be edited. To edit a document during a debug session, you must first stop the debug session.

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.



Print

Printer: \\printmaster\Prtba5

☐ Selection only

☐ Insert document name in header

☐ Insert date in header

☐ Page numbers

☐ Show line number every 10 lines

Number of copies: 1

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 83.

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 Function Libraries with an Application Area

You can associate an open function library with the currently open application area.

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 637.

To associate a 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 623.
3. Save the function library in your Quality Center project as an attachment. For more information, see “Saving a Function Library” on page 627.
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 adds [QualityCenter] to the file path and 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**.

Tip: You can remove an associated function library from the list by selecting it and clicking 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 83.

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 639.

2 Define the function, as described in "Defining the Function Definition" on page 641.

3 Register the function to a test object, if needed, as described in "Registering a Function Using the Function Generator" on page 642.

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 646.
5. Document the function by adding header information to it, as described in "Documenting the Function" on page 647.
6. Preview the function before finalizing it, as described in "Previewing the Function" on page 649.
7. Generate another function definition, if needed, as described in "Generating Another User-Defined Function" on page 649.
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 650.

Note: Each of the steps listed in this section assumes that you have performed the previous steps.

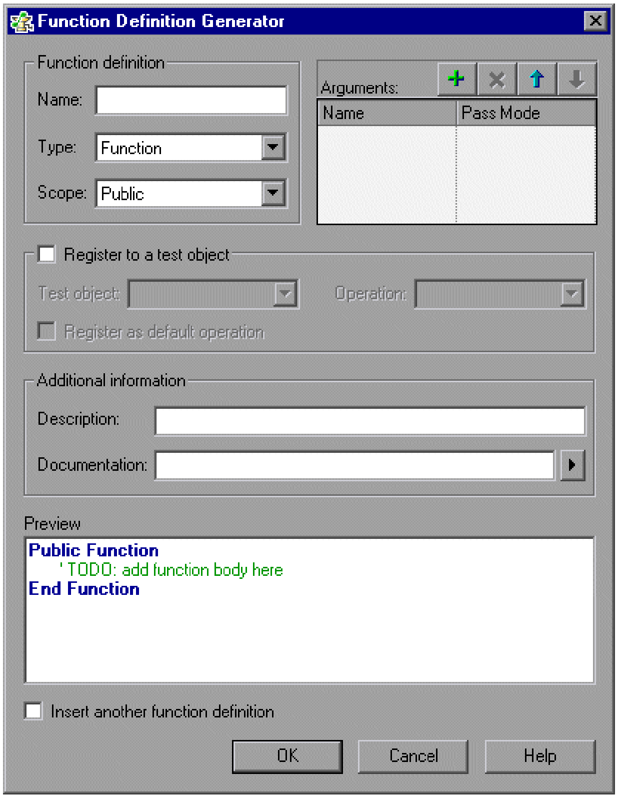
Opening the Function Definition Generator

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.



Name:

Type: Function

Scope: Public

☐ Register to a test object

Test object:

Operation:

☐ Register as default operation

Description:

Documentation:

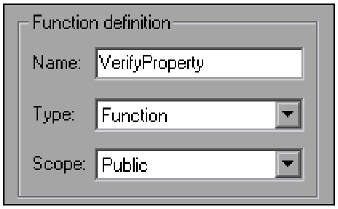
```
Public Function
    ' TODO: add function body here
End Function
```

☐ Insert another function definition

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.



Name: VerifyProperty

Type: Function

Scope: Public

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, refer to 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 646.

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.

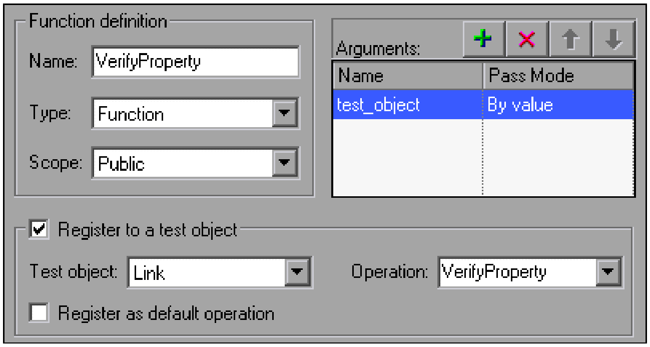
To define this function as the default function, you define the value of the fourth argument of the **RegisterUserFunc** statement as **True**. For example:

```
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.)



Function definition

Name: VerifyProperty

Type: Function

Scope: Public

Arguments:

Name	Pass Mode
test_object	By value

☑ Register to a test object

Test object: Link

Operation: VerifyProperty

☐ Register as default operation

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.

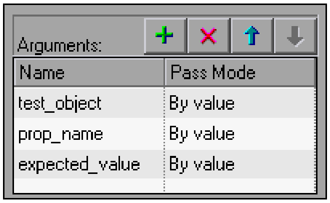
For example, if you were to define the **VerifyProperty** operation as the default operation for the Link test object, the value **True** would be defined as the fourth argument of the **RegisterUserFunc** statement, and the syntax would appear as follows:

```
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.



Name	Pass Mode
test_object	By value
prop_name	By value
expected_value	By value

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 642, 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 argument(s) 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 argument(s) 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).

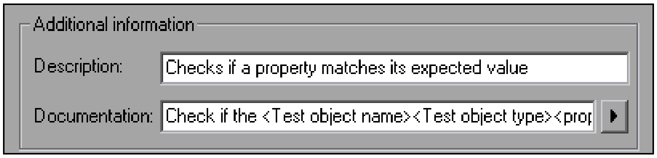
Documenting the Function

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 a clear and understandable.

For example, if you were checking a link to “Mercury” from a search engine, you might define the following documentation using the Function Definition Generator:

‘@Documentation Check if the <Test object name> <Test object type> <prop_name> value matches the expected value: <expected_value>.



Additional information

Description: Checks if a property matches its expected value

Documentation: Check if the <Test object name><Test object type><prop

After choosing values for the arguments in the Keyword View, the above documentation might appear as follows: Check if the “Mercury Business Technology” link “text” value matches the expected value: “Mercury 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.

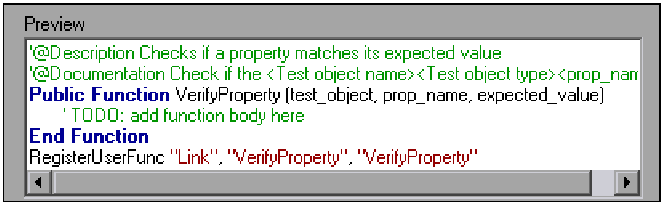
2 In the **Documentation** box, enter the text to be displayed in the **Documentation** column of the Keyword View. You can use arguments in the **Documentation** text by clicking ▸ and selecting the relevant argument.

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.



```
'@Description Checks if a property matches its expected value
'@Documentation Check if the <Test object name><Test object type><prop_nam
Public Function VerifyProperty (test_object, prop_name, expected_value)
    ' TODO: add function body here
End Function
RegisterUserFunc "Link", "VerifyProperty", "VerifyProperty"
```

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 641.

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 function(s). 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 635.

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 642, 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	**Note:** 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.

Item	Description
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. 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, refer to the *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**).

24

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.

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, refer to the *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 632.

Generating Statements in the Expert View or 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, refer to the *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," below.
- 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 663.

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.

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 11, "Customizing Function Library Windows."

When the **Statement Completion** option is enabled:

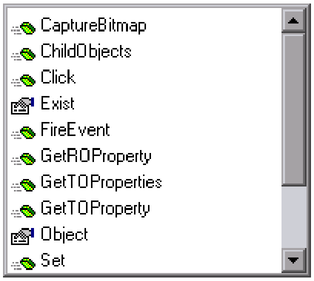
- If you type a period after a test object in a statement, QuickTest displays a list of the relevant methods, properties, and registered functions that you can add after the object you typed.
- If you type the name of a method or property, QuickTest displays a list of available methods and properties. 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 701.

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:

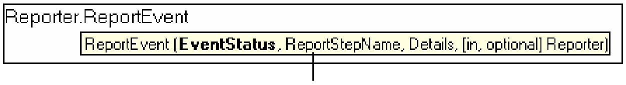
```
Browser("Welcome: Mercury Tours").Page("Book a Flight:
    Mercury).WebEdit("username").
```

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.



```
Reporter.ReportEvent
        ReportEvent (EventStatus, ReportStepName, Details, [in, optional] Reporter)
```

Statement completion 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, refer to the *QuickTest Professional Object Model Reference*.

For more information on VBScript syntax, see “Understanding Basic VBScript Syntax” on page 673.

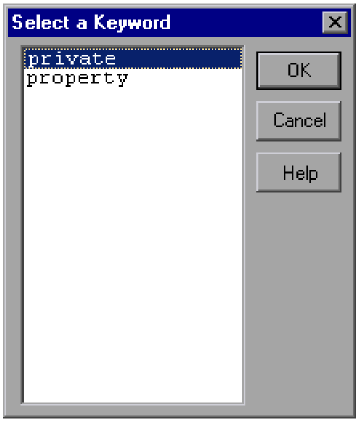
Automatically Completing VBScript Syntax

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 272.

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 673.

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

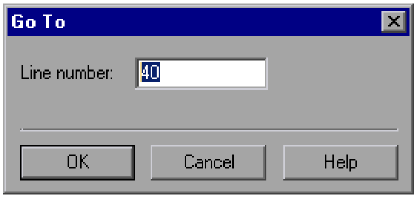
You can use the Go To dialog box to navigate to a specific line in a function library.

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 11, "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 Click the **Go To** button, or choose **Edit** > **Go To**. The Go To dialog box opens.



Line number: 40

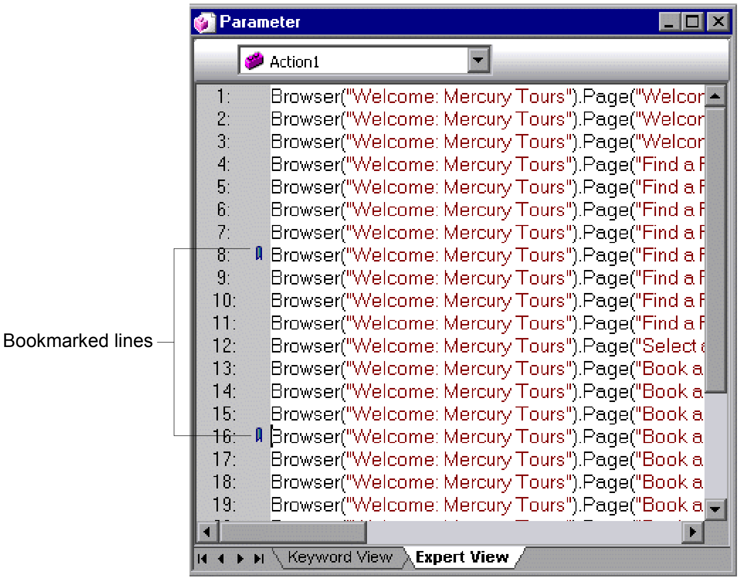
3 Enter the line to which you want to navigate in the **Line number** box and click **OK**. The cursor moves to the beginning of the line you specify.

Working with Bookmarks

You can use bookmarks to mark important sections in your function library so that you can easily navigate between the various parts. 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.



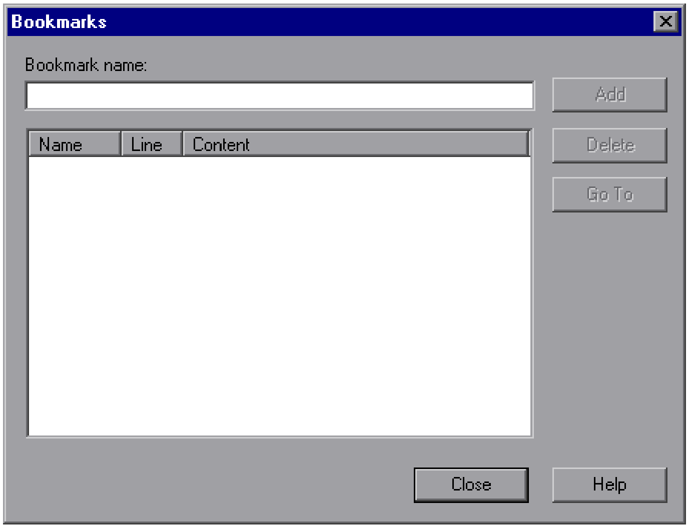
```
Browser("Welcome: Mercury Tours").Page("Welcor
Browser("Welcome: Mercury Tours").Page("Welcor
Browser("Welcome: Mercury Tours").Page("Welcor
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Find a F
Browser("Welcome: Mercury Tours").Page("Select
Browser("Welcome: Mercury Tours").Page("Book a
Browser("Welcome: Mercury Tours").Page("Book a
Browser("Welcome: Mercury Tours").Page("Book a
Browser("Welcome: Mercury Tours").Page("Book a
Browser("Welcome: Mercury Tours").Page("Book a
Browser("Welcome: Mercury Tours").Page("Book a
Browser("Welcome: Mercury Tours").Page("Book a
```

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.

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 11, "Customizing Function Library Windows."

Finding Text Strings

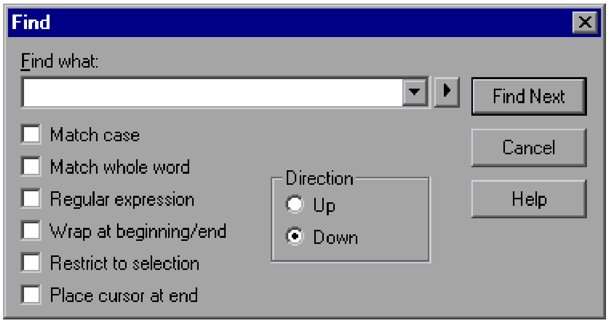
You can specify text strings to locate in a function library. You can either search for literal text or use regular expressions for a more advanced search. You can also use other options to further fine-tune your search results.

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.



Find

Find what:

☐ Match case
☐ Match whole word
☐ Regular expression
☐ Wrap at beginning/end
☐ Restrict to selection
☐ Place cursor at end

Direction
○ Up
● Down

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 672.

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.

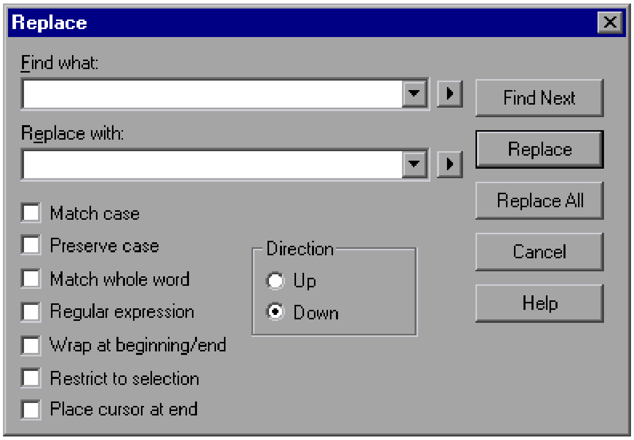
To replace a text string:

1 In the function library, perform one of the following:

➤ Click the **Replace** button.

➤ Choose **Edit** > **Replace**.

The Replace dialog box opens.



Find what:

Replace with:

☐ Match case

☐ Preserve case

☐ Match whole word

☐ Regular expression

☐ Wrap at beginning/end

☐ Restrict to selection

☐ Place cursor at end

Direction

○ Up

● Down

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 672.

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.

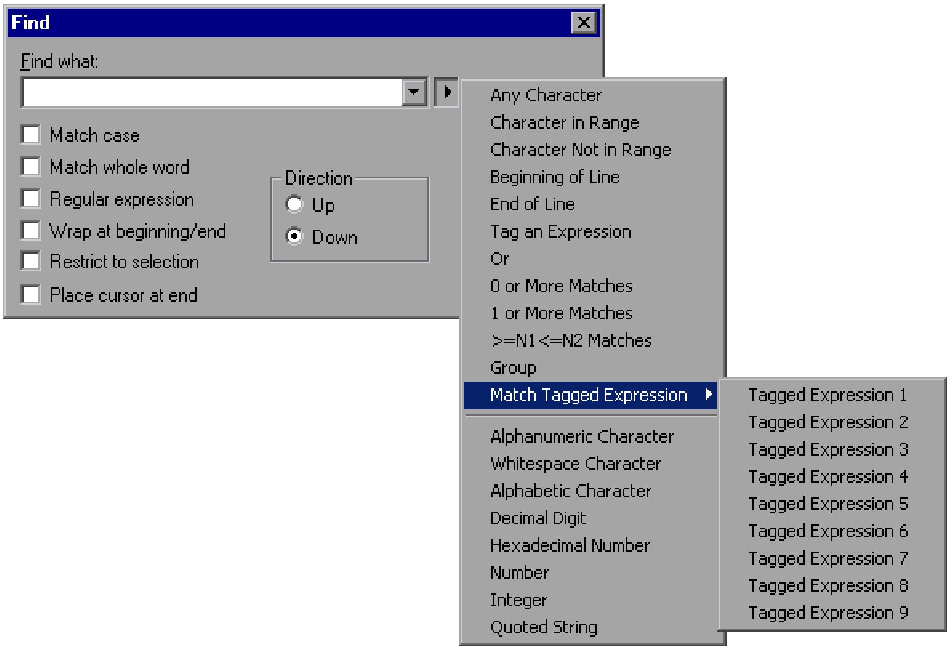
6 Click **Find Next** to highlight the next occurrence of the specified text string in the active function library.

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 352. 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.



Find

Find what:

□ Match case
□ Match whole word
□ Regular expression
□ Wrap at beginning/end
□ Restrict to selection
□ Place cursor at end

Direction
○ Up
● Down

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,” below.

- **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 676.

- **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 677.

- **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 677, and “Inserting Comments” on page 693.

- **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 693.

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 693.

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 statements and clicking the **Indent** button. Alternatively, you can select text and choose **Edit** > **Advanced** > **Indent** or 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 272.

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 the statement and clicking the **Outdent** button. Alternatively, you can choose **Edit** > **Advanced** > **Outdent** or you can delete the space at the beginning of the statements.

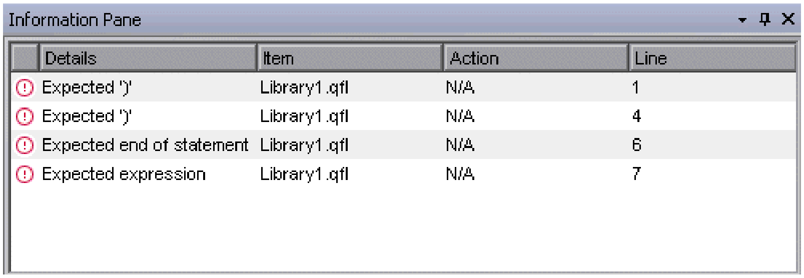
  For more detailed information on formatting in VBScript, you can view the VBScript documentation from the QuickTest Help menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).

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.



Details	Item	Action	Line
Expected ')'	Library1.qfl	N/A	1
Expected ')'	Library1.qfl	N/A	4
Expected end of statement	Library1.qfl	N/A	6
Expected expression	Library1.qfl	N/A	7

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'. **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 you record an operation on an object, QuickTest 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 was recorded in a browser with the name Mercury Tours.

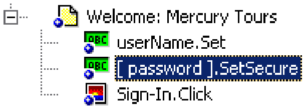
```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username")
```

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 Web site or 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.

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 14, “Working with Test 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, refer to the **Utility** section of the *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
```

Tip: When creating a programmatic description for an ActiveX test object and the relevant run-time object is windowless (has no window handle associated with it), you must add the **windowless** property to the description and set its value to **True**.

For example:

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

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, refer to the *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.

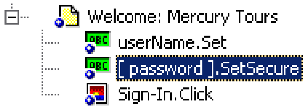
Once you have “built” a description in your description object, use the following syntax to retrieve child objects that match the description:

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, refer to the *QuickTest Professional Object Model Reference*.

Using Programmatic Descriptions for the WebElement Object

The **WebElement** object enables you to perform methods on Web objects that may not fit into any other Mercury test object class. The **WebElement** test object is never recorded, but you can use a programmatic description with the **WebElement** object to perform methods on any Web object in your Web site.

For example, when you run the statement below:

```
Browser("Mercury Tours").Page("Mercury Tours").
    WebElement("Name:=UserName", "Index:=0").Click
```

or

```
set WebObjDesc = Description.Create()
WebObjDesc("Name").Value = "UserName"
WebObjDesc("Index").Value = "0"
Browser("Mercury Tours").Page("Mercury Tours").WebElement(WebObjDesc).
   Click
```

QuickTest clicks on the first Web object in the Mercury Tours page with the name UserName.

For more information on the WebElement object, refer to the *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.

Creating Checkpoints Programmatically

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 702. For information on using **While...Wend** statements, see “While...Wend Statement” on page 698. For information on specific test objects, methods, and properties, refer to the *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, refer to the *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, refer to the *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 276.

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, refer to 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 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. **Default = 1**. **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} condition]
    statement
Loop
```

Item	Description
condition	A condition to be fulfilled.
statement	A statement or series of statements to be performed during the loop.

In the following example, 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 condition2 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("Mercury Technologies").Page("Mercury Technologies").
    Link("Jobs").GetROProperty("href")
```

Tip: If you do not know the test object properties of objects in your Web site or application, you can view them using the Object Spy. For information on the Object Spy, see Chapter 13, "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, refer to the *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 the Expert View or a Function Library" on page 660.

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 703.

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, refer to the *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, refer to the *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 680.

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, refer to 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://msdn.microsoft.com/library/en-us/winprog/winprog/windows_api_start_page.asp?frame=true

A reference to specific API functions can be found at: http://msdn.microsoft.com/library/en-us/winprog/winprog/windows_api_reference.asp?frame=true

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 value(s).

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, refer to the *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
men_hwnd = Extern.GetMenu(hwin)' Get window's main menu's handle
MsgBox men_hwnd
' Use API Functions
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, refer to the *QuickTest Professional Object Model Reference*.

25

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.

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 easily 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 recording on 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.

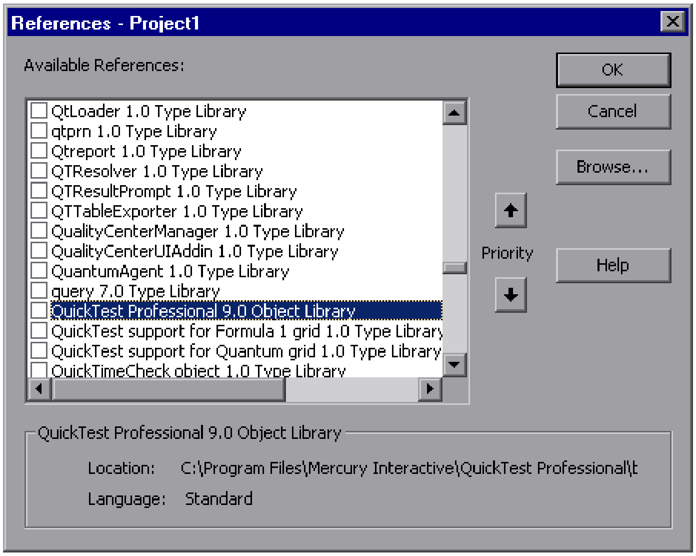
Writing Your Automation Script

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).



References - Project1

Available References:

- ☐ QtLoader 1.0 Type Library
- ☐ qtprn 1.0 Type Library
- ☐ Qtreport 1.0 Type Library
- ☐ QTResolver 1.0 Type Library
- ☐ QTResultPrompt 1.0 Type Library
- ☐ QTTableExporter 1.0 Type Library
- ☐ QualityCenterManager 1.0 Type Library
- ☐ QualityCenterUIAddin 1.0 Type Library
- ☐ QuantumAgent 1.0 Type Library
- ☐ query 7.0 Type Library
- ☐ QuickTest Professional 9.0 Object Library
- ☐ QuickTest support for Formula 1 grid 1.0 Type Library
- ☐ QuickTest support for Quantum grid 1.0 Type Library
- ☐ QuickTimeCheck object 1.0 Type Library

QuickTest Professional 9.0 Object Library

Location: C:\Program Files\Mercury Interactive\QuickTest Professional\t

Language: Standard

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, refer to 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, refer to the online *QuickTest Automation Object Model Reference*. For more information on this Help file, see "Using the QuickTest Automation Reference" on page 716.

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 9, “Setting Global Testing Options” and Chapter 15, “Configuring Object Identification.”see Chapter 9, “Setting Global Testing Options,” Chapter 15, “Configuring Object Identification,” and Chapter 26, “Setting Options for Individual Tests.”

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 VII

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:

Recording and Running Components

- **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 Chapter 12, "Configuring Web Event Recording.") QuickTest can then run the component by running the steps as they originally occurred.

- **How can I record on objects or environments not supported by QuickTest?**

  You can do this in a number of ways:

  - By default, QuickTest supports several developmental environments. You can also enable support for additional environments, such as Java, Oracle, .NET, SAP Solutions, Siebel, PeopleSoft, terminal emulators, and Web services, by installing and loading any of the external add-ins that are available for QuickTest Professional.
  - 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 411.

- **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\Mercury Interactive\QuickTest Professional\bin\QTPro.exe
  ```

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.

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 23, "Working with User-Defined Functions and Function Libraries."

➤ **How do I make the component prompt the user for input while it is running?**

You can use the VBScript **InputBox** function 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, refer to the *VBScript Reference*.

The following example shows the **InputBox** function used to prompt the user for a password.

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set
"administrator"
Passwd = InputBox ("Enter password", "User Input")
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("password").Set
Passwd
```

Working with Dynamic Content

- **How can I record and run components on objects that change dynamically from viewing to viewing?**

  Sometimes the content of objects in a Web page or application changes due to dynamic content. You can create dynamic descriptions of these objects so that QuickTest will recognize them when it runs the test. For more information, see Chapter 14, "Working with Test Objects."

- **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:

  ```
  Browser("Window_name").Exist
  ```

  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.

  For more information on the **Exist** property and **ChildObjects** method, refer to the *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.

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 browser can be accessed through the browser's Document Object Model (DOM). 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?**

  Dynamically created Web pages utilizing Active Server Page technology have an .asp extension. This technology is completely server-side and has 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, refer to the *QuickTest Professional User's Guide*.

- **How can I access HTML tags directly?**

  QuickTest provides direct access to the browser's Document Object Model (DOM) through which you can access the HTML tags directly. Access to the DOM is performed using the .Object notation.

  The test below demonstrates how to iterate over all the tags in a page. The test 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 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 IE Document Object Model?**

  For information on the IE DOM, browse to the following Web sites:

  Document object:
  http://msdn.microsoft.com/workshop/author/dhtml/reference/objects/obj_document.asp

  Other DHTML objects:
  http://msdn.microsoft.com/workshop/author/dhtml/reference/objects.asp

  General DHTML reference:
  http://msdn.microsoft.com/workshop/author/dhtml/reference/dhtml_reference_entry.asp

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, refer to the *QuickTest Professional User's 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. When your application changes, you can re-record part of a component. If the change is not significant, you can manually edit a component to update it.

  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.

To update the information about your test object properties when object properties change, use the **Update Run Mode** option. For more information, see “Updating a Component” on page 533.

- **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 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:

- Do not load unnecessary add-ins in the Add-in Manager when QuickTest starts. This will improve both recording time and run session performance. For more information on loading add-ins, see “Loading QuickTest Add-ins” on page 199.
- 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 test without displaying the execution arrow for each step, enabling the test to run faster. For more information on the Run tab of the Options dialog box, see “Setting Run Testing Options” on page 220.

Index