Astra QuickTest™

User's Guide

Version 5.5

MERCURY INTERACTIVE

Astra QuickTest 5.5 User's Guide

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

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089 USA

AQTUG5.5/01

Table of Contents

PART VI: CONFIGURING QUICKTEST

PART VII: WORKING WITH OTHER MERCURY TOOLS

PART VIII: SUPPORTED ENVIRONMENTS

Welcome to QuickTest

Welcome to QuickTest, Mercury Interactive's functional testing tool for Web applications. QuickTest provides everything you need to quickly create and run tests.

Using This Guide

This guide describes how to use QuickTest to test your Web or Windows applications. It provides step-by-step instructions to help you create, debug, run tests, and report defects detected during the testing process.

It contains 8 parts:

Part I Starting the Testing Process

Provides an overview of QuickTest and the main stages of the testing process.

Part II Working With Test Objects

Explains how QuickTest identifies objects in your application.

Part III Creating Tests

Describes how to create tests, insert checkpoints, assign parameters, use regular expressions, actions, and handle unexpected events that occur during a test run.

Part IV Running and Debugging Tests

Describes how to run tests and analyze test results, and how to control test runs to identify and isolate bugs in test scripts.

Part V Advanced Features

Describes how to enhance your test in Expert View mode and introduces several programming techniques to create a more powerful test. It also describes how to streamline the testing process of your Web or Windows applications. **This section is recommended for advanced users of QuickTest.**

Part VI Configuring QuickTest

Describes how to change QuickTest's default settings, both globally and per test. It also describes how to customize the test script editor.

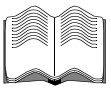
Part VII Working with Other Mercury Tools

Describes how QuickTest interacts with TestDirector, Mercury Interactive's test management tool, and how you can run tests and call functions in compiled modules from WinRunner, Mercury Interactive's enterprise functional testing tool for Microsoft Windows applications.

Part VIII Supported Environments

Includes information for testing ActiveX controls, Java applets and applications, and multimedia applications.

QuickTest Documentation Set



In addition to this user's guide, QuickTest comes with the following printed documentation:

QuickTest Installation Guide explains how to install QuickTest on a single computer or on a network.

QuickTest Tutorial teaches you basic QuickTest skills and shows you how to start testing your applications.

QuickTest Object Model Reference provides access to the QuickTest VBScript methods, including a description of each object, a list of the methods associated with each object, description syntax, and an example of usage for each object and method.

Online Resources

QuickTest includes the following online resources:

Read Me First (available from the QuickTest Start menu program group) provides last-minute news and information about QuickTest.

What's New in QuickTest (available from the QuickTest Help menu) describes the newest features and supported environments in the latest versions of QuickTest.

QuickTest Tutorial (available from the QuickTest Welcome window or the QuickTest Help menu) teaches you basic QuickTest skills and shows you how to start testing your applications.

Books Online (available from the QuickTest Start menu program group or the QuickTest Help menu) displays context-sensitive help for dialog boxes, the *QuickTest Object Model Reference*, and the *QuickTest User's Guide* in an HTML Help browser.

QuickTest Context-Sensitive Help (available from dialog boxes and windows) describes QuickTest dialog boxes and windows.

QuickTest Object Model Reference (available from the Help menu) gives you online access to the VBScript methods, including a description of each object, a list of the methods associated with each object, and description syntax and an example of usage for each object and method.

Mercury Tours sample Web site (available from the QuickTest Browser Settings dialog box) is the basis for many examples in this book. The URL for this Web site is *http://mercurytours.mercuryinteractive.com*.

VBScript Reference (available from the QuickTest Help menu) describes Microsoft's VBScript language, and includes a tutorial and function reference.

Technical Support Online (available from the QuickTest Help menu) uses your default Web browser to open Mercury Interactive's Customer Support Web site. This site enables you to browse the knowledge base and add your own articles, post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. The URL for this Web site is *http://support.mercuryinteractive.com*.

Support Information (available from the QuickTest Help menu) presents the locations of Mercury Interactive's Customer Support Web site and home page, the e-mail address for sending information requests, the name of the relevant news group, the location of Mercury Interactive's public FTP site, and a list of Mercury Interactive's offices around the world.

Mercury Interactive on the Web (available from the QuickTest Help menu) uses your default Web browser to open Mercury Interactive's home page. This site provides you with the most up-to-date information on Mercury Interactive and its products. This includes new software releases, seminars and trade shows, customer support, educational services, and more. The URL for this Web site is *http://www.mercuryinteractive.com*.

Typographical Conventions

This book uses the following typographical conventions:

1, 2, 3	Bold numbers indicate steps in a procedure.
➤	Bullets indicate options and features.
>	The greater than sign separates menu levels (for example, **File** > **Open**).
Stone Sans	The **Stone Sans** font indicates names of interface elements on which you perform actions (for example, "Click the **Run** button.").
Bold	**Bold** text indicates method or function names.
Italics	*Italic* text indicates method or function arguments, file names or paths and book titles.
<>	Angle brackets enclose a part of a file path or URL address that may vary from user to user (for example, *<MyProduct installation folder>\bin*.
Helvetica	The Helvetica font is used for examples and text that is to be typed literally.
[ ]	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

Starting the Testing Process

1

Introduction

Welcome to QuickTest, Mercury Interactive's functional testing tool for Web and Windows applications.

QuickTest enables you to test standard Web objects and ActiveX controls. In addition to these environments, QuickTest Professional also enables you to test Java applets and applications and multimedia objects on Web pages.

This guide provides you with detailed descriptions of QuickTest features and testing procedures.

Testing with QuickTest

QuickTest facilitates creating tests on your Web or Windows application by recording as you navigate. As you navigate through your site or application, QuickTest records each step you perform and generates a test that graphically displays this step in an icon-based *test tree*. For example, clicking a link, selecting a check box, or submitting a form are all recorded in your test.

In addition, you can instruct QuickTest to check the properties of specific objects in your site. For example, you can instruct QuickTest to check that a specific text string is displayed in a particular location on your Web page, or you can check that a hypertext link goes to the correct URL address.

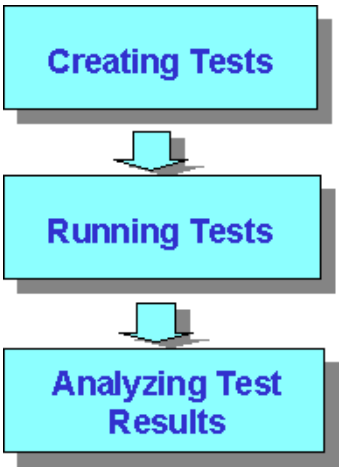
After you record, you can further enhance your test by adding and modifying steps in the test tree. When you run the test, QuickTest connects to your site or application and performs each step in your test. After you run your test, you can view a report detailing which steps in your test succeeded or failed.

Note that by default, each test includes a single *action*. An action is the basic unit in your test. You can divide your test into multiple actions in order to organize your test. Most of the chapters in this guide provide information on how to work with a single action. For information on why and how to work with multiple actions in a test, see Chapter 14, "Working with Actions."

Once you test the validity of your test in the Virtual User Recorder, you incorporate it in a load testing scenario. You use the Astra LoadTest Controller to run load tests and analyze your Web application's performance under load. Refer to the *Astra LoadTest Controller User's Guide* for information about load testing scenarios.

Testing Process

Testing with QuickTest involves 3 main stages:



Creating Tests

You create a test by recording a session on your site or application to check its functionality.

To create a test:

➤ Record a session on your site or application.

As you navigate through your site or application, QuickTest graphically displays each *step* you perform in the form of a collapsible icon-based *test tree*. A step is something that causes or makes a change in your site or application, such as clicking a link or image, or submitting a data form. For more information, see Chapter 5, "Creating Tests."

➤ Insert checkpoints into your test.

A *checkpoint* searches for a specific value of a page, object or text string and enables you to identify whether or not your Web site or application is functioning correctly. For more information, see Chapter 6, "Creating Checkpoints."

➤ Broaden the scope of your test by replacing fixed values with parameters.

When you test your site or application, you can *parameterize* your test to check how your application performs the same operations with different data. You may supply data in a table in the Data pane, or by defining environment variables and values, or you can have QuickTest generate random numbers or current user and test data. When you parameterize your test, QuickTest substitutes the parameters in your test with parameter values. When you use data table parameters, QuickTest uses the values from a different row in the data table during each *iteration* of the test. For more information, see Chapter 10, "Parameterizing Tests."

You can also use output values to extract data from your test. An *output value* is a value retrieved during the test run, and entered into your table in the Data pane. You can subsequently use this output value as an input variable in your test. This enables you to use data retrieved during a test in other parts of the test. For more information, see Chapter 11, "Creating Output Values."

Running Tests

After you create your test, you run it to check the behavior of your Web site.

You can:

➤ Run your test to check your site or application.

The test runs from the first line in your test and stops at the end of the test. While running, QuickTest connects to your Web site or application and performs each operation in your test, checking any text strings, objects or tables you specified. If you parameterized your test, QuickTest repeats the test for each set of data values you defined. For more information, see Chapter 17, "Running Tests."

➤ Run a test to debug your test.

You can control your test run to help you identify and eliminate defects in your test. You can use the *Step Into*, *Step Over*, and *Step Out* commands to run your test step by step. You can also set *breakpoints* to pause your test at pre-determined points. You can view the value of variables in your test each time the test stops at a breakpoint in the Debugger Views. For more information, see Chapter 19, "Debugging Tests."

Analyzing Test Results

After you run your test, you can view the test results.

You can:

➤ View the test results in the Test Results window.

After you run your test, the Test Results window opens and displays the results of your test. You can view a summary of your test results or a detailed report. For more information, see Chapter 18, "Analyzing Test Results."

➤ Report defects detected during a test run.

If you have TestDirector installed, you can report the defects you discover to a database. TestDirector is Mercury Interactive's software test management tool. For more information, see Chapter 28, "Working with TestDirector."

Expert View

You can use the Expert View tab to view a text-based version of your test. The test script is composed of VBScript statements (Microsoft's Visual Basic Scripting language) that correspond to the steps and checks displayed in your test tree. For more information, see Chapter 22, "Testing in the Expert View."

Actions

You can divide your test into sections called "actions." This enables you to build a robust and modular test composed of logical units that can be shared between tests. It also enables you to use an action in multiple tests by copying or calling the action from another test. For more information on parameterizing tests, see Chapter 10, "Parameterizing Tests." For more information on working with actions, see Chapter 14, "Working with Actions."

Sample Site

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

The first page of the Mercury Tours site is the login page. You must log in to begin using the site. To log in, enter "mercury" as your member name and "mercury" as your password.

Managing the Testing Process

QuickTest works with TestDirector, Mercury Interactive's test management tool. You can use TestDirector to create a project (central repository) of manual and automated tests, build test cycles, run tests, and report and track defects. You can also create reports and graphs to help you review the progress of test planning, test runs, and defect tracking before a software release.

When you work in QuickTest, you can create and save tests directly to your TestDirector project. You can also run QuickTest tests from TestDirector and then use TestDirector to review and manage the results. For more information, see Chapter 28, "Working with TestDirector."

2

QuickTest at a Glance

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

This chapter describes:

- Starting QuickTest
- The QuickTest Window
- Test Pane
- Display Pane
- Data Pane
- Debugger Pane
- Using QuickTest Commands
- Loading QuickTest Add-Ins

Starting QuickTest

To start QuickTest, click **Programs** > **QuickTest** > **QuickTest** in the Start menu.

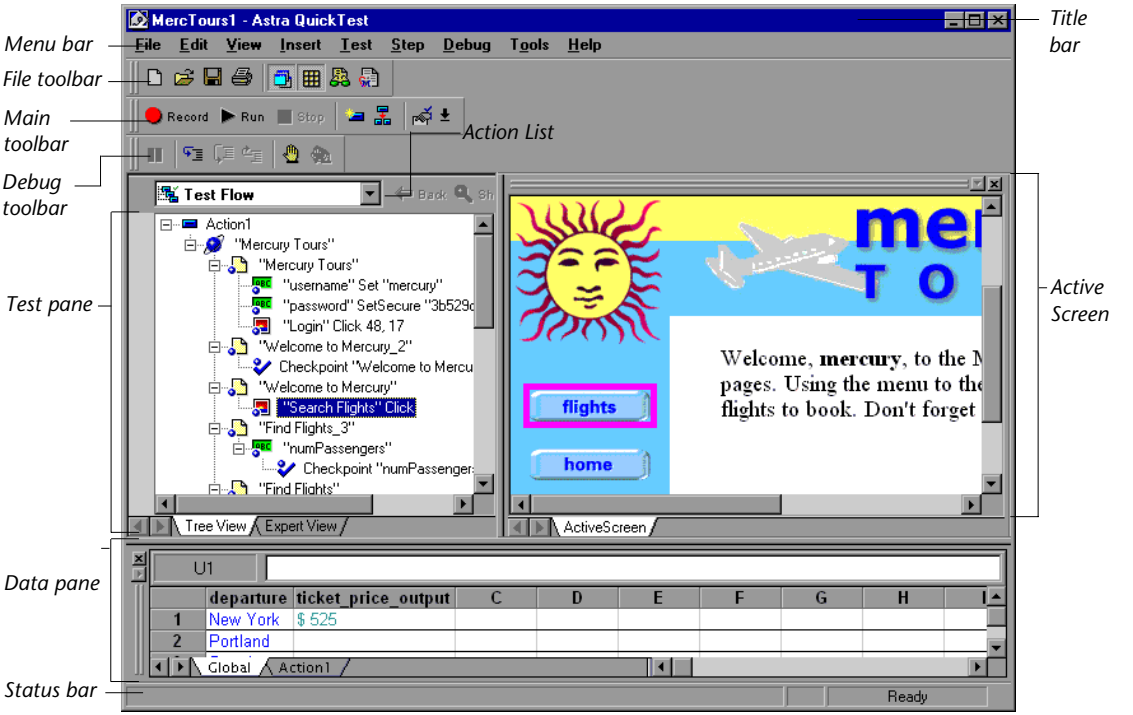
The first time you start QuickTest, the Welcome to QuickTest window opens. You can choose to open the QuickTest tutorial, start recording a new test, open an existing test, or close the welcome window to begin working in a new test.

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

The QuickTest Window

The QuickTest window contains the following key elements:

- *QuickTest title bar,* displaying the name of the currently open test
- *Menu bar,* displaying menus of QuickTest commands
- *File toolbar,* containing buttons to assist you in managing your test
- *Main toolbar,* containing buttons to assist you in the testing process
- *Debug toolbar,* containing buttons to assist you in debugging your test
- *Action toolbar*, containing buttons and a list of actions, enabling you to view the details of an individual action or the entire test flow
- *Test pane*, containing two tabs to view your test—the Tree View and the Expert View
- *Display pane*, containing the ActiveScreen
- *Data pane*, containing two tabs to assist you in parameterizing your test—Global and Action
- *Debugger pane*, containing three tabs to assist you in debugging your test—Watch Expressions, Variables, and Command.
- *Status bar*, displaying the status of the test

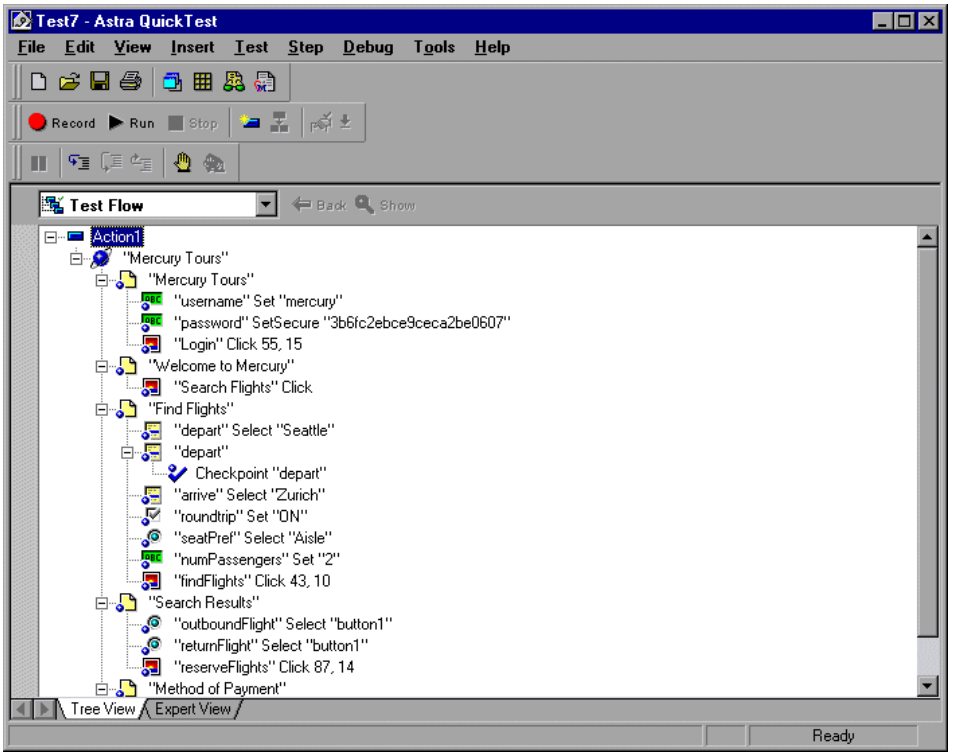

MercTours1 - Astra QuickTest
File Edit View Insert Test Step Debug Tools Help
Record Run Stop
Test Flow
Action1
"Mercury Tours"
"Mercury Tours"
"username" Set "mercury"
"password" SetSecure "3b529c
"Login" Click 48, 17
"Welcome to Mercury_2"
Checkpoint "Welcome to Mercu
"Welcome to Mercury"
"Search Flights" Click
"Find Flights_3"
"numPassengers"
Checkpoint "numPassenger
"Find Flights"
Tree View
Expert View
flights
home
Welcome, mercury, to the
pages. Using the menu to th
flights to book. Don't forget
ActiveScreen
U1
departure
ticket_price_output
C
D
E
F
G
H
1
New York
$ 525
2
Portland
Global
Action1
Ready
Title bar
Menu bar
File toolbar
Main toolbar
Debug toolbar
Action List
Test pane
Active Screen
Data pane
Status bar

Test Pane

The Test pane contains two tabs to view your test—Tree View and Expert View.

Tree View Tab

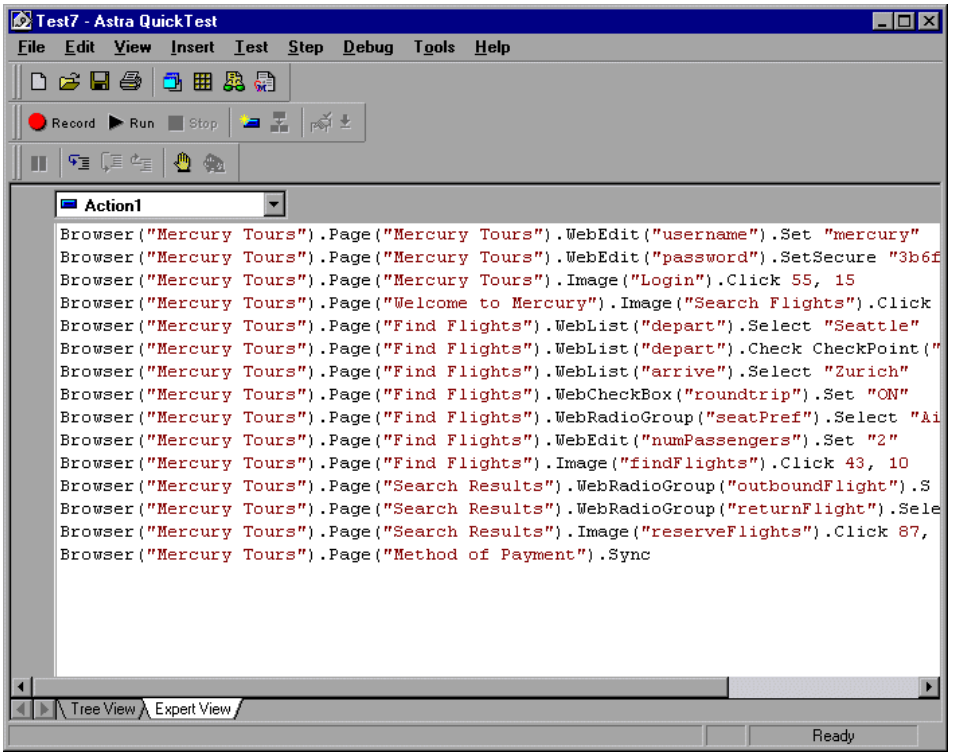
In the Tree View tab (default mode), QuickTest displays your test in the form of a collapsible icon-based *test tree*. Each operation performed on your Web-based application is recorded as an icon in your test tree.



For every icon in the Tree View, QuickTest displays a corresponding line of script in the Expert View.

Expert View Tab

In the Expert View tab, QuickTest displays each action in your test in the form of a script instead of an icon tree. Your script is composed of VBScript statements. For every statement in the Expert View tab, a corresponding icon exists in the test tree in the Tree View tab. For more information on using the Expert View, see Chapter 22, "Testing in the Expert View."



Display Pane

QuickTest's Display pane contains the ActiveScreen. To view this pane, click the **ActiveScreen** button or choose **View** > **ActiveScreen**.

The ActiveScreen displays the page or object corresponding to a highlighted step in your test. It provides you with an easy way to view your test, make modifications, and add checkpoints.

Data Pane

In a new test, the Data pane contains two tabs to assist you in parameterizing your test—Global and Action1. If you add new actions to your test, corresponding action tabs are added to the data pane. To view this pane, click the **Data Views** button or choose **View** > **Data Views**.

Global Tab

The Global tab contains variable values for the parameters defined in your parameterized test. The variable values are available for an entire test. When you run your parameterized test, QuickTest inserts the data from the Global tab into the test. For more information, see Chapter 14, “Working with Actions.”

Action Tab

The Action tab contains variable values for the parameters defined in your parameterized test. The variable values are available only for a specific action and not for an entire test. When you run your parameterized test, QuickTest inserts the data from the Action tab into the relevant action in the test. For more information, see Chapter 14, “Working with Actions.”

Debugger Pane

The Debugger pane contains three tabs to assist you in debugging your test—Watch Expressions, Variables, and Command. To view the Debugger pane, click the **Debug View** button or choose **View** > **Debug View.**

Watch Expressions Tab

The Watch Expressions tab enables you to view the current value of any variable or VBScript object that you enter in the Watch Expressions table.

Variables Tab

The Variables tab enables you to view the current value of all variables that have been recognized up to the breakpoint where the test stopped.

Command Tab

The Command tab enables you to execute a line of script in order to set or modify the current value of a variable or VBScript object in your test. When you continue running the test, QuickTest uses the new value that was set in the command.

For more information on using the Debugger pane, see Chapter 19, "Debugging Tests."

Using QuickTest Commands

You can select QuickTest commands from the menu bar or from a toolbar. Certain QuickTest commands can also be executed by pressing shortcut keys.

Choosing Commands on a Menu

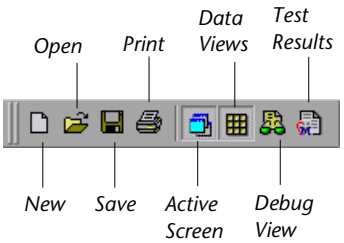
You can choose all QuickTest commands from the menu bar.

Clicking Commands on a Toolbar

You can execute some QuickTest commands by clicking buttons on the toolbars. QuickTest has four built-in toolbars: the *File toolbar*, the *Main toolbar*, and the *Debug toolbar*, and the *Action toolbar*.

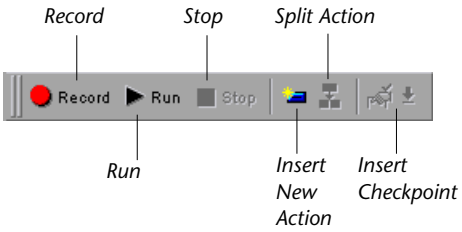
File Toolbar

The File toolbar contains buttons for managing a test. For more information on managing your test, see Chapter 5, "Creating Tests." The following buttons are displayed on the File toolbar:



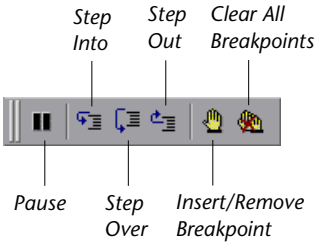
Main Toolbar

The Main toolbar contains buttons for the commands used when creating and maintaining your test. The following buttons are displayed on the Main toolbar:



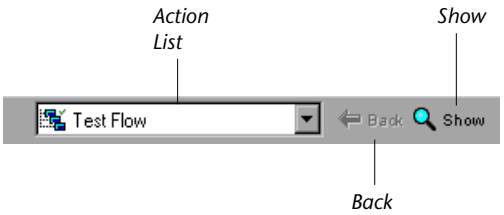
Debug Toolbar

The Debug toolbar contains buttons for the commands used when debugging the steps in your test. The following buttons are displayed on the Debug toolbar:



Action Toolbar

The Action toolbar contains options that enable you to view all actions in the test flow or to view the details of a selected action. The following options are displayed on the Action toolbar:



When you have reusable or external actions in your test, the Action toolbar is always visible. If there are no reusable or external actions in your test, you can choose to show or hide the Action toolbar in the General Tab of the Options dialog box.

When you have reusable or external actions in your test, only the action icon is visible when viewing the entire Test Flow in the Tree View. You can view the details of the reusable or external actions by double-clicking on the action, selecting the action name from the list in the Action toolbar, or selecting the action in the tree and clicking the Show button. You can return to the Test Flow by clicking the Back button.

Executing Commands Using Shortcut Keys

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

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

Command	Shortcut Key	Function
New	CTRL + N	Creates a new test and closes your browser.
Open	CTRL + O	Opens a test.
Save	CTRL + S	Saves the active test.
Export to Zip File	CTRL + ALT + S	Creates a zip file of the active test.
Import from Zip File	CTRL + ALT + O	Imports a test from a zip file.
Print	CTRL + P	Prints the active test.

You can execute 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 (Expert View only).
Redo	CTRL + Y	Reverses the action of the Undo command (Expert View only).
Cut	CTRL + X	Removes the selection from your test.
Copy	CTRL + C	Copies the selection from your test.
Paste	CTRL + V	Pastes the selection to your test.
Delete	DEL	Deletes the selection from your test.
Rename Action	F2	Changes the name of an action (Tree View only).
Find	CTRL + F	Searches for a specified character (Expert View only).
Replace	CTRL + H	Searches and replaces a specified character (Expert View only).
Go To	CTRL + G	Moves to a particular line in the test (Expert View only).
Complete Word	CTRL + SPACE	When you type the beginning of a VBScript method or object, completes the word (Expert View only).
Parameter Info	CTRL + SHIFT + SPACE	Displays the syntax of a parameter (Expert View only).

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

Command	Shortcut Key	Function
Standard Checkpoint	F12	Creates a standard checkpoint for an object or a table.
Standard Output Value	CTRL + F12	Creates a standard output value for a text string, an object, or a table.

You can execute the following Test menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
Record	F3	Starts recording mode.
Run	F5	Runs the test from the beginning or from the line at which the test was stopped.
Stop	F4	Stops test recording or the test run.
Low Level Recording	CTRL + SHIFT + F3	Starts low level recording.

By pressing the ALT + ENTER shortcut keys, you can execute the following Step menu commands depending on the selected test tree node:

Command	Selected test tree node	Function
Object Properties	Test Object	Opens the Object Properties dialog box.
Action Properties	Action	Opens the Action Properties dialog box.
Method Arguments	containing a method	Opens the Method Arguments dialog box.
Checkpoint Properties	Checkpoint	Opens the checkpoint dialog box corresponding to the selected checkpoint.
Output Value Properties	Output Value	Opens the output value dialog box corresponding to the selected output value.

Note: You can use the shortcut keys for the Step menu commands only in the Tree View.

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

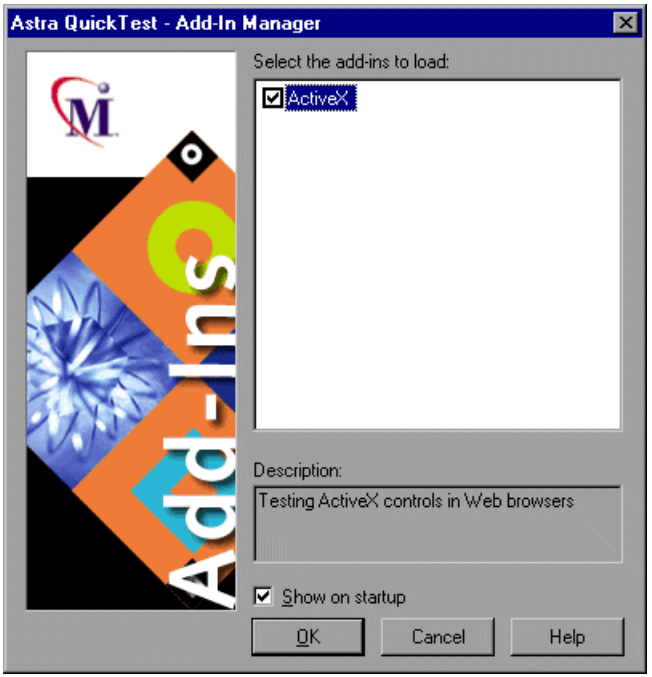
Command	Shortcut Key	Function
Pause	PAUSE	Stops the test run after the statement has been executed. The test run can be resumed from this point.
Step Into	F11	Runs only the current line of the test script. If the current line calls a method, the method is displayed in the view but is not executed.
Step Over	F10	Runs only the current line of the test script. When the current line calls a method, the method is executed in its entirety, but is not displayed in the view.
Step Out	SHIFT + F11	Runs to the end of the method then pauses execution. (Available only after running a method using Step Into.)
Insert/Remove Breakpoint	F9	Sets or clears a breakpoint in the test.
Clear All Breakpoints	CTRL + SHIFT + F9	Deletes all breakpoints in the test.

You can also execute commands from the Data Table menu using shortcut keys. For more information about Data Table menu commands, see "Editing the Data Table," on page 265.

Loading QuickTest Add-Ins

If you installed the ActiveX add-in while installing QuickTest or afterward, you can specify to load the add-in at the beginning of each QuickTest session.

When you start QuickTest, the **Add-In Manager** dialog box opens. It displays a list of all installed add-ins for QuickTest. You can select which add-ins to load for the current session of QuickTest.



The first time QuickTest is started, by default, no add-ins are selected. At the beginning of each subsequent QuickTest session, your selection from the previous session is the default setting.

You can hide the Add-In Manager dialog box by clearing the **Show on startup** check box. By using the **Show Add-In Manager on startup** check box in the **General** tab of the **Options** dialog box, you can display it again. For information on working with the Options dialog box, see Chapter 24, “Setting Global Testing Options.”

Part II

Working With Test Objects

3

Understanding the Test Object Model

This chapter explains the concepts of Test Object and Runtime Object, describes how QuickTest identifies objects in your application and 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.

This chapter describes:

- Understanding the Test Object and Runtime Object Concepts
- Understanding How QuickTest Learns Objects
- Viewing Object Properties Using the Object Spy
- Viewing Object Methods and Method Syntax Using the Object Spy

About the Test Object Model

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 test to represent the actual object in your application and to store information about that object that will help it identify and check the object during the test run.

A *runtime object* is the actual object in your Web site or application on which methods are performed during the test run.

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

- identifies the QuickTest test object class that represents the object on which you performed the operation and creates the appropriate test object in the test.
- reads the current value of the object's properties in your application and stores the list of properties and values with the test object.
- records the operation that you performed on the object using the appropriate QuickTest test object method.

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

```
<INPUT TYPE="submit" NAME="Submit" VALUE="Find">
```

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

Property	Value
type	submit
name	Find
index	0
html tag	INPUT

It also records that you performed a **Click** method on the WebButton.

When you run a test, 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. For example, QuickTest would look for a WebButton object with the HTML tag: INPUT, of type: submit, with the name: Find. When it finds the object, it performs the **Click** method on it.

Understanding the Test Object and Runtime Object Concepts

Throughout this guide, you will see the terms *Test Object* and *Runtime Object*. The following principles should help you distinguish between these two concepts.

- *Test objects* are the objects in your test that represent the objects in your Web site or application. *Runtime objects* are the objects in your application during the test run.
- The *test object* properties set is created and maintained by QuickTest. The *runtime object* property set is created and maintained by the object creator (Microsoft for Internet Explorer objects, Netscape for Netscape objects, the product developer for ActiveX objects, etc.)

  Similarly, *test object* methods are methods that QuickTest recognizes and records when they are performed on an object while you are recording a test, and that QuickTest performs when your test runs. *Runtime object* methods are the methods of the object in your application as defined by the object creator. You can access and perform *runtime object* methods using the **Object** property.

  For information about activating runtime methods using the "Object" property, see "Accessing Runtime Object Properties and Methods" on page 374.
- Each *test object* method you perform while recording a test is recorded as a separate step in your test. When you run your test, QuickTest performs the recorded *test object* method on the *runtime object*.
- *Test object* properties are the properties whose values are captured from the objects in your Web site or application when you record your test. QuickTest uses the values of these properties to identify *runtime objects* in your application during a test run.
- Property values of objects in your application may change dynamically each time your application opens, or based on certain conditions. In order to make the *test object* property values match the property values of the *runtime object*, you can modify *test object* properties manually while designing your test or with **SetTOProperty** statements during a test run, you can parameterize them with data table parameters so that a different value will

be used during each iteration of the test, or you can use regular expressions to identify property values based on conditions or patterns you define.

For more information on modifying object properties, see Chapter 4, "Managing Test Objects."

For more information on parameterization, see Chapter 10, "Parameterizing Tests."

For more information on regular expressions, see Chapter 12, "Using Regular Expressions."

- You can view or modify the *test object* property values that are stored with your test in the Object Properties or Object Repository dialog boxes. 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 about the Object Properties and Object Repository dialog boxes, see "Modifying Test Object Properties While Designing Your Test," on page 37.

  For information about viewing *test object* property values using the Object Spy, see Chapter 3, "Viewing Object Properties Using the Object Spy."

- You can view the syntax of the *test object* methods as well as the runtime methods of any object on your desktop using the Methods tab of the Object Spy.

  For information about viewing the methods of an object using the Object Spy, see "Viewing Object Methods and Method Syntax Using the Object Spy" on page 33.

- You can retrieve or modify current *test object* property values during the test run by adding **GetTOProperty** and **SetTOProperty** statements in the Expert View. You can retrieve *runtime object* property values during the test run by adding statements that use the **Object** property.

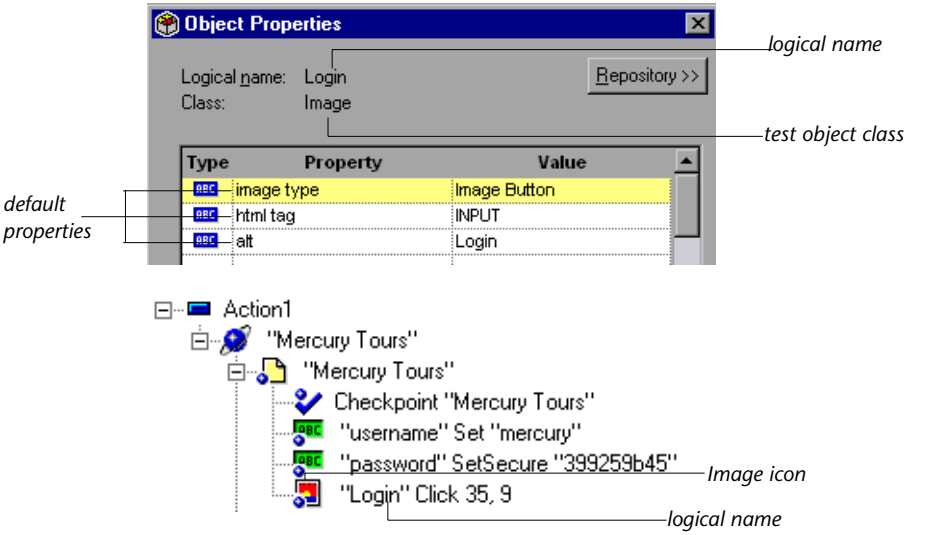
  For information about retrieving runtime property values using the **Object** property, see "Accessing Runtime Object Properties and Methods" on page 374.

  For information about the **GetTOProperty** and **SetTOProperty** methods and the Object property, refer to the *QuickTest Object Model Reference*.

Understanding How QuickTest Learns Objects

QuickTest learns the values of an object's default properties when it records a test, and it uses this information to identify the object when it runs the test.

For each object class, QuickTest learns a set of default properties, which it uses to identify objects of that class in your application. For example, by default, QuickTest identifies an image based on its logical name, the image type (such as plain image or image button), the HTML tag, and the alternate text (if any).



Viewing Object Properties Using the Object Spy

Using the Object Spy, you can view the properties of any object on your desktop. You use the Object Spy pointer to point to an object. The Object Spy displays the object hierarchy tree and the properties and values of the selected object in the Properties tab of the Object Spy dialog box.

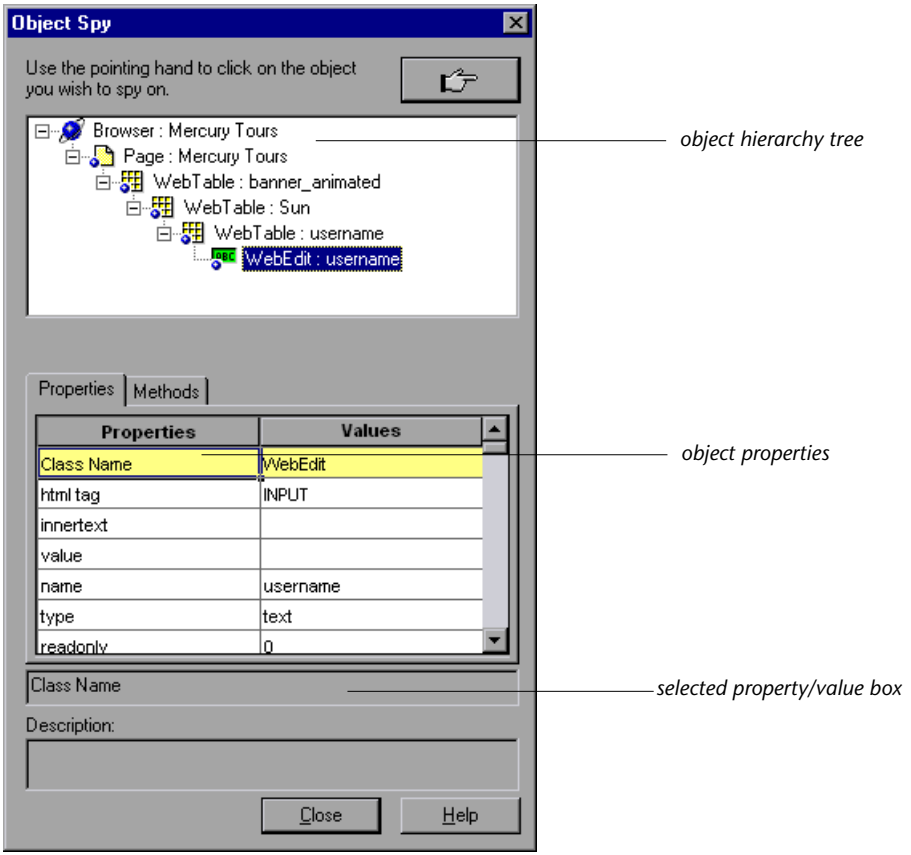
To view object properties:

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

2 Choose **Tools** > **Object Spy** to open the Object Spy dialog box.

3 Click the pointing hand. Both QuickTest and the Object Spy are minimized so that you can point to any object in the open application.

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



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

6 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 and use CTRL-C to copy the text to the clipboard or right-click the highlighted text and choose **Copy** from the menu.

Note: If the value of a property contains more than one line, the value cell of the object properties list indicates: "multi-line value". To view the value, click the value cell. The selected property/value box displays the value with delimiters indicating the line breaks.

Viewing Object Methods and Method Syntax Using the Object Spy

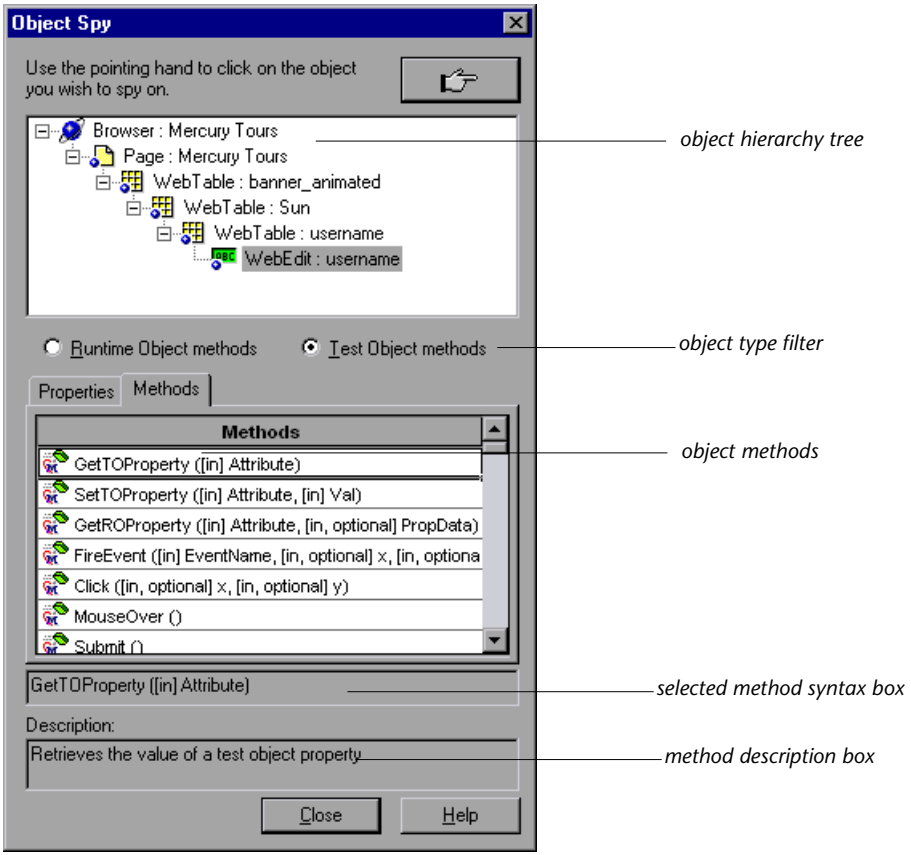
In addition to viewing object properties, the Object Spy also enables you to view both the runtime object methods and the test object methods associated with an object and to view the syntax for a selected method. You use the Object Spy pointer to point to an object. The Object Spy displays the object hierarchy tree and the runtime 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** to open the Object Spy dialog box.
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.
5. Click the object for which you want to view the associated methods. The Object Spy returns to focus and displays the object hierarchy tree and the

runtime object or *test object* methods associated with the object that is selected within the tree.



6 To view the methods of the test object, click the **Test Object methods** radio button. To view the methods of the runtime object, click the **Runtime Object methods** radio button.

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

8 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 and use CTRL-C to copy the text to the clipboard, or right-click the highlighted text and choose **Copy** from the menu.

4

Managing Test Objects

This chapter explains how to manage and maintain the test objects in your test. 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. It also describes how to delete an object from your test.

This chapter describes:

- About Managing Test Objects
- Understanding the Object Repository and Object Properties Dialog Boxes
- Modifying Test Object Properties While Designing Your Test
- Working with Test Objects During a Test Run
- Understanding Dynamic Descriptions of Objects
- Modifying How QuickTest Identifies Objects
- Deleting an Object from the Object Repository

About Managing Test Objects

QuickTest identifies objects in your Web site or application based on a set of default test object properties. It stores the object data it learns in the Object Repository.

If one or more of the property values of an object in your application differ from the property values stored with the test object, your test will fail because QuickTest will be unable to identify the object. Thus, when objects in your Web site or application change, you must modify the corresponding test object properties so that you can continue to use your existing tests.

There are several methods for doing this. You must choose the method that best fits your needs:

- You can manually change a test object property value to match a new static property of an object in your application.
- You can modify the set of properties that QuickTest uses to identify the object, so that it will be able to identify an object even when some of its properties change dynamically.
- You can use the **SetTOProperty** method to modify object properties during a test run.
- You can parameterize a test object property with a data table parameter if you expect the property value to change in a predictable way with each iteration of the test.
- You can use regular expressions to identify an object based on conditions or patterns you define.

You can make most of these modifications in the Object Properties dialog box or in the Object Repository dialog box. Changes you make to an object affect all occurrences of that object within a given action.

Changes you make using the **SetTOProperty** method only affect the object property during the test run, and do not affect the property values in the Object Repository.

This chapter includes information on the first three options above. For more information on parameterizing object properties, see Chapter 10, "Parameterizing Tests." For information on using regular expressions, see Chapter 12, "Using Regular Expressions."

Understanding the Object Repository and Object Properties Dialog Boxes

The *Object Repository* dialog box displays all objects in the current action in a test tree. You can use the Object Repository dialog box to view or modify the properties of any test object in the selected action. The Object Repository stores objects on a per-action basis. Thus, if the same test object is used in several steps within the same action, you need to modify the object's properties only one time. If the object is used again in another action, however, you need to update the object's properties in that action as well. For more information about actions, see Chapter 14, "Working with Actions."

Even when steps containing a test object are deleted from your test script, the objects remain in the Object Repository. If you delete all occurrences of an object from a test action, you can also choose to delete the object from the Object Repository.

The *Object Properties dialog box* accesses test object information from the Object Repository, and displays the properties and values of the test object in the selected step.

Modifying Test Object Properties While Designing Your Test

As the content of your Web site or application changes, you can continue to use tests you developed previously. Your test includes all steps you perform, such as clicking hypertext and image links, typing in edit fields, and clicking buttons. As Web sites and applications change, the objects in the steps may also change.

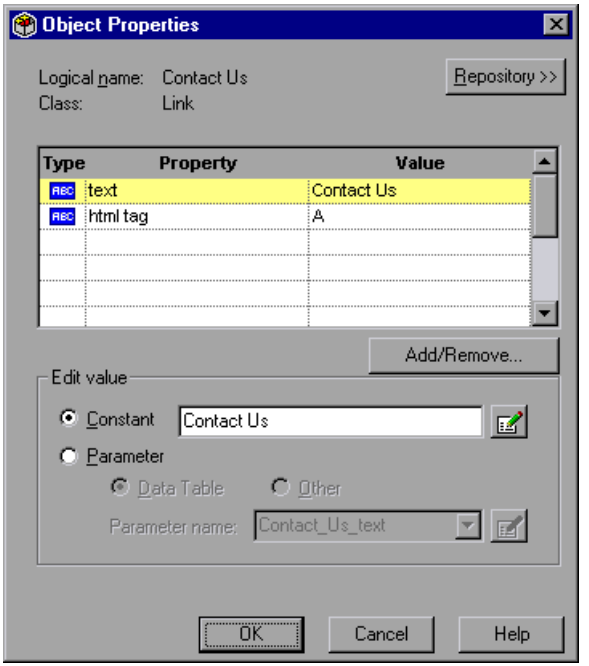
Suppose an object in your test changes. You would need to modify the step in your test containing the object so that QuickTest can continue to identify it. For example, if the MyCompany Web site has a "Contact Us" hypertext link and then the text string in this link is changed to "Contact MyCompany." You need to update your test so that QuickTest will continue to identify the link properly.

You can modify the object by modifying one or more of the object's property values in the Object Repository or Object Properties dialog box.

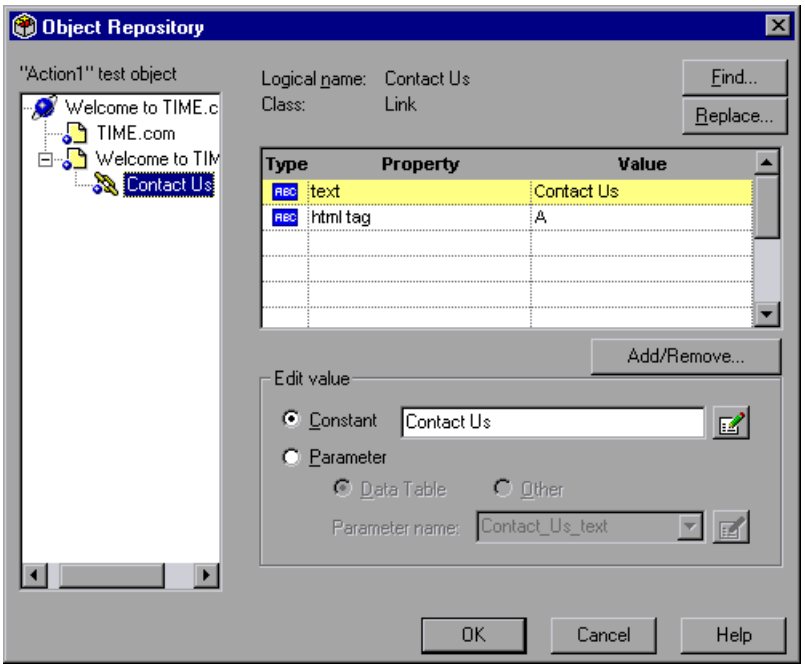
To modify an object property in your test:

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

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



If you want to view all objects in the action, click the **Repository** button. The Object Repository dialog box opens and displays the test tree of the action.



Tip: You can also open the Object Repository for the selected action by choosing **Tools** > **Object Repository**.

---

2 Highlight the property and value to modify.

3 In the **Constant** box, enter a new value for the property.

4 If you want to set the property value as a regular expression, click the **Edit Constant Value Options** button. For information about using regular expressions, see "Using Regular Expressions in Steps," on page 204.

Note: You can also create a data table parameter and set the values in the data table as regular expressions. For more information about parameterizing your test, see Chapter 10, "Parameterizing Tests."

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

Finding Test Object Properties

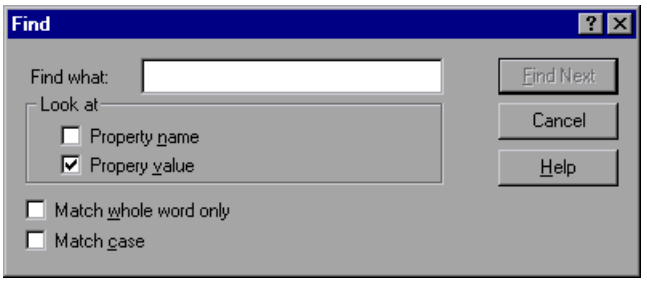
You can use the **Find** and **Replace** buttons in the Object Repository dialog box to find or find and modify a property or value that occurs several times in the same action.

To find a property or value in the Object Repository:

1 Right-click an action containing an object with the property or value you want to find and choose **Object Properties,** and then click the **Repository** button, or choose **Tools** > **Object Repository**.

The Object Repository dialog box opens.

2 Right-click the object in the repository tree and choose **Find**, or click the **Find** button. The Find dialog box opens.



3 Enter the text for the property or value you want to find. Select **Property name**, **Property value**, or both.

➤ If you want the search to find only words that exactly match the text you entered, select **Match whole word only**.

➤ If you want the search to distinguish between upper and lower case letters, select **Match case**.

Click **Find Next**. The first instance of the searched word is displayed.

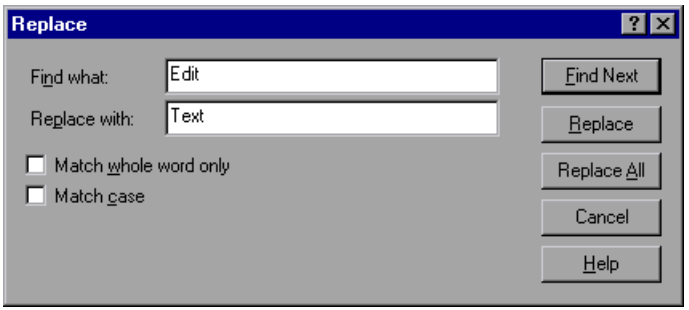
4 To find the next instance, click **Find Next** again.

To find and replace a value in the Object Repository:

1 Right-click an action containing an object with the property or value you want to find and choose **Object Properties,** and then click the **Repository** button, or choose **Tools** > **Object Repository**.

The Object Repository dialog box opens.

2 Right-click the object in the repository tree and choose **Find and Replace** or click the **Replace** button. The Replace dialog box opens.



3 Enter the text for the property value you want to find.

➤ If you want the search to find only words that exactly match the text you entered, select **Match whole word only**.

➤ If you want the search to distinguish between upper and lower case letters, select **Match case**.

4 To individually find and replace each instance of the word(s) you are searching for one at a time, click **Find Next**. When an instance is found, click **Replace**. Then click **Find Next** again to find the next instance.

Note: You cannot replace property names. You also cannot replace values in a read-only test or action.

Tip: To replace all instances of the word you are searching for with the new value in a single action, click **Replace All**.

Working with Test Objects During a Test Run

As QuickTest performs each step in your test, it creates a live instance of the test object in the step. For recorded steps, QuickTest uses the properties in the Object Repository to create the live instance of the object. For the remainder of the test, QuickTest refers to the live instance of the test object rather than to the test object in the Object Repository.

You can also create live instances of test objects to represent objects from your Web site or application without using the Object Repository by using descriptive programming.

Modifying Test Object Properties During a Test Run

You can modify the properties of the live instance of the object during the test run without affecting the permanent values in the object repository by adding a **SetTOProperty** statement to in the Expert View.

Use the following syntax for the SetTOProperty method:

object(*"description"*).**SetTOProperty** *Property, Value*

For more information, refer to the *QuickTest Object Model Reference*.

Creating Test Objects During a Test Run

Descriptive programming enables you to create live instances of test objects to represent objects from your Web site or application and to 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 add a statement in the Expert View that enters a value in the new edit box, rather than recording a new step in an existing test.

For more information on descriptive programming, see Chapter 22, "Testing in the Expert View."

Understanding Dynamic Descriptions of Objects

You can change the properties that QuickTest uses to identify an object. This is useful when you want to create and run tests on an object that changes dynamically. An object may change dynamically if it is frequently updated or if it is created using dynamic content, e.g. from a database. You can also change the properties that identify an object in order 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 news letters. The archive page includes a hypertext link to the current news letter and to all past news letters. The text in the first hypertext link on the page changes as the current news letter changes, but it always links to a page called *current.html*. Suppose you want to create a step in your test 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, 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.

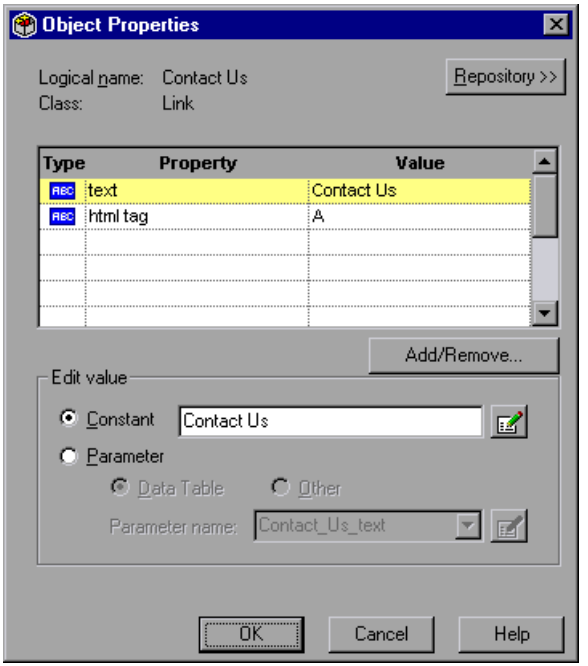
Modifying How QuickTest Identifies Objects

The Object Properties dialog box contains the properties of a given object. Alternatively, the Object Repository contains a list of all objects in an action, their properties, and their values. By default, QuickTest learns certain properties for each object class. You can use the Add/Remove Properties dialog box to instruct QuickTest to learn properties of the object other than the default properties.

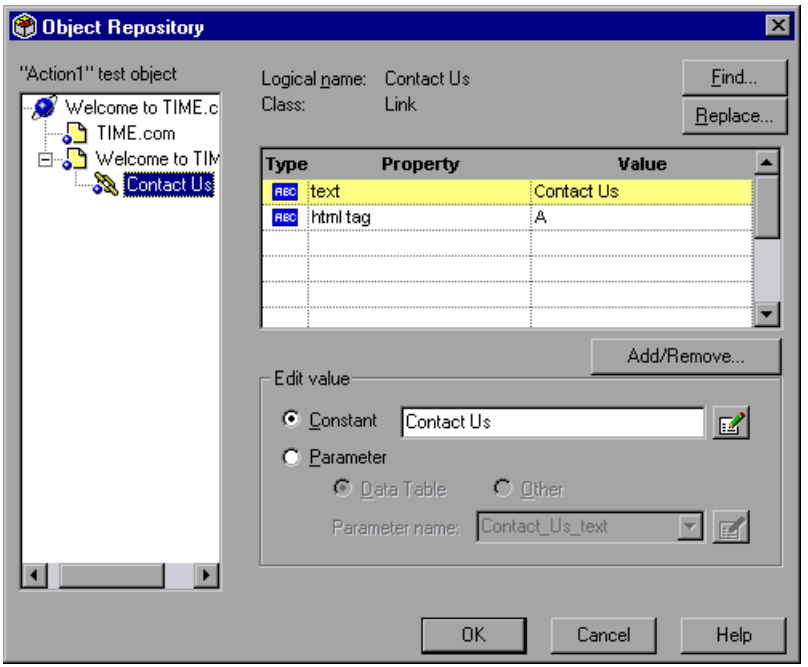
To modify how QuickTest identifies an object:

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

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



If you want to view all objects in the action, click the **Repository** button. The Object Repository dialog box opens and displays the test tree of the action.

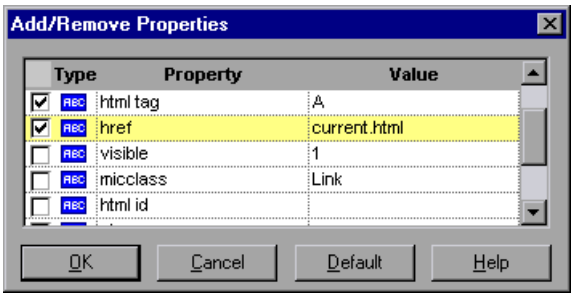


Tip: You can also open the Object Repository for the selected action by choosing **Tools** > **Object Repository**.

---

2 Click the **Add/Remove** button.

The Add/Remove Properties dialog box opens, listing the properties that can be used to identify the object. A selected check box next to a property indicates a property that QuickTest uses to identify the object. The value for each property is displayed in the Value column.



3 Modify the default properties that QuickTest uses to identify the object:

- To add a property, select the corresponding check box.
- To remove a property, clear the corresponding check box.

4 Click **OK** to close the Add/Remove Properties dialog box.

The Object Repository or Object Properties dialog box is reactivated.

5 Click **OK** to save and close the Object Repository or Object Properties dialog box.

Tip: After you add a new property, you can modify its value in the Edit Value box in the Object Repository or Object Properties dialog box. For more information on modifying object properties, see "Modifying Test Object Properties While Designing Your Test," on page 37.

Deleting an Object from the Object Repository

When you remove a step from an action in your test script, the object remains in the Object Repository. If the object in the step you removed does not occur in any other steps within that action, you can delete the object from the Object Repository.

Note: If your action contains references to an object that you have deleted from the Object Repository, your test will not run.

To delete an object from the Object Repository:

1. Right-click an action containing an object with the property or value you want to find and choose **Object Properties,** and then click the **Repository** button, or choose **Tools** > **Object Repository**.

   The Object Repository dialog box opens.

2. In the repository tree, right-click the step containing the object you want to delete and click **Delete**. A confirmation message is displayed.

3. Click **Yes** to confirm that you want to delete the object. The object is deleted from the Object Repository.

Note: Once you confirm that you want to delete the object, you cannot retrieve the object again. Clicking **Cancel** after confirming the deletion does not cancel the deletion.

Part III

Creating Tests

5

Creating Tests

You can quickly create a test by recording the operations you perform on your Web site or application.

This chapter describes:

- Planning a Test
- Recording a Test
- Creating Checkpoints
- Understanding Your Test
- Changing the ActiveScreen
- Managing a Test

About Creating Tests

QuickTest enables you to generate an automated test by recording the typical processes that you perform on your Web site or application. As you navigate through your application, QuickTest graphically displays each *step* you perform as an icon in a *test tree*. A step is anything a user does that changes the content of a page in your site or application, for example, clicking a link, or typing data into an edit box.

While recording or designing your test, you can insert checkpoints into your test. A *checkpoint* compares the value of an element captured in your test when you record your test with the value of the same elements captured during the test run. This helps you determine whether or not your Web site or application is functioning correctly.

When you test your site, you may want to check how it performs the same operations with different data. This is called *parameterizing* your test. You may supply data in a table in the Data pane, or by defining environment variables and values, or you can have QuickTest generate random numbers or current user and test data. When you parameterize your test, QuickTest substitutes the parameters in your test with parameter values. When you use data table parameters, QuickTest uses the values from a different row in the data table during each *iteration* of the test. For more information, see Chapter 10, "Parameterizing Tests."

After recording, you can further enhance your test by adding and modifying steps in the test tree.

Planning a Test

Before you start recording, you should plan your test. You should consider the following:

- Determine the functionality you want to test. Short tests that check specific functions of the site or complete a transaction are better than long tests that perform several tasks.
- Decide which information you want to check during the test. A checkpoint can check for differences in the text strings, objects, and tables in your site. For more information, see Chapter 6, "Creating Checkpoints."
- Evaluate the types of events you need to record. If you want to record more or fewer events than QuickTest generally records by default, you can configure the events you want to record. For more information, see Chapter 20, "Configuring Event Recording."
- Consider increasing the power and flexibility of your test by replacing fixed values with parameters. When you parameterize your test, you can check how it performs the same operations with multiple sets of data. For more information, see Chapter 10, "Parameterizing Tests."

- You can change the way that QuickTest identifies objects. This is particularly helpful when your application contains objects that change frequently or are created using dynamic content, e.g. from a database. For additional information, see Chapter 4, "Managing Test Objects."
- If you are an advanced user, consider using actions to streamline the testing process. For additional information, see Chapter 14, "Working with Actions."
- You can link to WinRunner tests and call WinRunner TSL functions from an QuickTest test. For additional information, see Chapter 29, "Working with WinRunner."

Recording a Test

You create a test by recording the typical processes that users perform. QuickTest records each step you perform and generates a test tree.

Note that by default, each test includes a single action, but a test can include multiple actions. This chapter describes how to record a test with a single action. For information on why and how to work with multiple actions, see Chapter 14, "Working with Actions."

By default, QuickTest records in the standard recording mode. If you are unable to record on an object in a given environment in the standard recording mode, or if you want to record mouse clicks and keyboard input with the exact x- and y-coordinates, you may want to record on those objects using low-level recording. This section describes how to record a test in the standard recording mode. For more information about low-level recording, see "Recording and Running Tests," on page 377.

Tip: If you have objects that behave like standard objects, but are not recognized by QuickTest, you can define your objects as virtual objects rather than using the low-level recording mode. For more information, see Chapter 13, "Learning Virtual Objects."

Consider the following guidelines when recording a test:

- Before you start to record, close all applications not required for the test.
- Determine the security zone of a Web site. When you record a test on a Web browser, the browser may prompt you with security alert dialog boxes. You may choose to disable/enable these dialog boxes.
- You can control how QuickTest records and displays your tests by setting testing options in the Options dialog box. For more information, see Chapter 24, "Setting Global Testing Options."
- If you are creating a test on Web objects, you can record your test on one browser and run it on another browser. QuickTest supports the following browsers: Netscape Navigator, Microsoft Internet Explorer, and AOL. For additional information, see Chapter 7, "Checking Web Objects."

To record a test:

1 Open QuickTest. For more information, see "Starting QuickTest" on page 10.

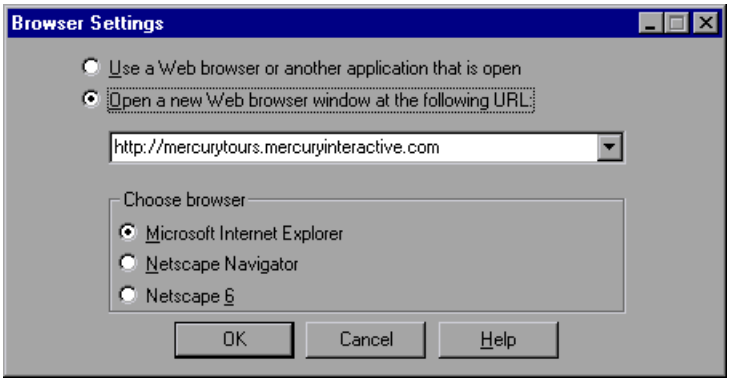
2 Open a test:

- To create a new test, click the **New** button or choose **File** > **New**.

- To open an existing test, click the **Open** button or choose **File** > **Open**. In the **Open QuickTest Test** dialog box, select a test and click **Open**.

For more information, see "Managing a Test" on page 59.

3 Click the **Record** button or choose **Test** > **Record**. The Browser Settings dialog box opens.



- If this is your first recording session in a test, choose which browser to use and specify a URL. By default, the URL for the Mercury Tours site is displayed for the address.
- If this is not your first recording session in a test, QuickTest remembers your choices from the previous session. Proceed to step 5.

4 Choose which browser to use and whether to use an existing Web browser window or to open a new browser window to a specified location.

Note: QuickTest supports Microsoft Internet Explorer, Netscape Navigator, and the AOL browser. For information on which versions of these browsers are supported, refer to the *ReadMe* file.

If you select **Open a new Web browser window at the following URL**, type or select the Web address from which you want to start recording the test. By default, the URL for the Mercury Tours site is displayed as the address.

Click **OK**. If you chose to open a new Web browser, then it opens, displaying the Web address you specified.

Notes: If you choose to use an existing Web browser window, then you must start the browser session after starting QuickTest.

The options you select in the Browser Settings dialog box also set the startup settings for the test. For more information about startup settings, see "Browser Test Settings," on page 407.

---

5 Navigate through your Web site or application. QuickTest records each step you make in the test tree in the Tree View tab.

6 You can insert text checkpoints, object checkpoints, and table checkpoints to compare the values of the specified property during a test run with the values stored for the test object property within the test, in order to determine whether or not a site is functioning correctly. For more information, see Chapter 6, "Creating Checkpoints."

7 You can parameterize your test to check how it performs the same operations with multiple sets of data. For more information, see Chapter 10, "Parameterizing Tests."

8 When you complete your recording session, click the **Stop** button or choose **Test** > **Stop**.

9 To save your test, click the **Save** button or choose **File** > **Save**. In the Save QuickTest Test dialog box, assign a name to the test. For more information, see "Managing a Test," on page 59.

Creating Checkpoints

QuickTest enables you to add checkpoints to your test. A *checkpoint* is a step in your test that compares the values of the specified property during a test run with the values stored for the same test object property within the test. This enables you to identify whether or not your Web site or application is functioning correctly.

You can create checkpoints to check various objects in a Web site or application. You can create checkpoints for Web, ActiveX, Java, and multimedia objects.

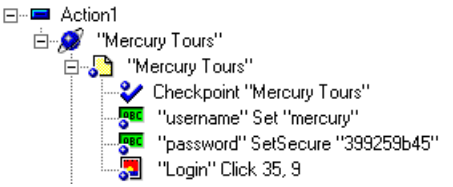
- **Web objects:** Create checkpoints on Web page properties, Web accessibility support, text strings, tables, images and form elements. You can also use formulas to check that the data displayed on a Web page is valid. For more information, see Chapter 7, "Checking Web Objects."
- **ActiveX controls:** Create checkpoints on ActiveX controls. For more information, see Chapter 30, "Testing ActiveX Controls."
- **Java objects:** Create checkpoints on Java applets, applications, or objects. For more information, see Chapter 31, "Testing Java Applets and Applications". Note that you can test Java applets, applications, and objects only in QuickTest Professional.
- **Multimedia objects:** Create checkpoints on Macromedia Flash, Windows MediaPlayer, and RealPlayer multimedia objects. For more information about checking multimedia objects, see Chapter 32, "Testing Multimedia Applications". Note that you can test multimedia objects only in QuickTest Professional.

For general information about creating checkpoints, see Chapter 6, "Creating Checkpoints."

Understanding Your Test

While recording, QuickTest creates a *test tree*—a graphical representation of the navigation you perform on your site. The test tree is displayed in the Tree View tab. Each step in the tree represents a step performed on your site and browser.

The following is a sample test of a login procedure to the Mercury Tours site, Mercury Interactive's sample Web site.



The table below provides an explanation of each step in the tree.

Step	Description
Action1	*Action1* is the action name.
"Mercury Tours"	The browser invokes the *Mercury Tours* site.
"Mercury Tours"	The name of the Web page.
Checkpoint "Mercury Tours"	A checkpoint that checks statistical information including the load time, the links and the source of images in the *Mercury Tours* page.
"username" Set "mercury"	*username* is the name of the edit box. *Set* is the method performed on the edit box. *mercury* is the value of the edit box.
"password" SetSecure "399259b45"	*password* is the name of the edit box. *SetSecure* is an encryption method performed on the edit box. *399259b45* is the encrypted value of the password.
"Login" Click 35, 9	*Login* is the name of the image. *Click* is the method performed on the image. *35, 9* are the x- and y-coordinates where the image was clicked.

Changing the ActiveScreen

As the content of your Web site or application changes, you can continue to use tests you developed previously. You simply change the ActiveScreen display so that QuickTest can continue to find the objects in your modified site.

For example, suppose that one of the pages in the Mercury Tours site now includes a new object and you want to add a checkpoint that checks for this object. You can use the Change ActiveScreen command to replace the page in your ActiveScreen tab and then proceed to create a checkpoint for this object.

To change the ActiveScreen:

1. Make sure that your Web browser displays the page that you want to use to replace with the current ActiveScreen tab display.
2. In the test tree, click a step that you want to change, the page is displayed in the ActiveScreen tab.
3. Choose **Tools** > **Change ActiveScreen**. The QuickTest window is minimized, and the mouse pointer becomes a pointing hand.
4. Click the page displayed in your browser. A message prompts you to change your current ActiveScreen display.
5. Click **Yes**.

Managing a Test

You can use the File toolbar to create, open, save, zip, and print recorded tests.

Creating a New Test

To create a new test, click the **New** button or choose **File** > **New**. A new test opens. You are ready to start recording your test.

Opening an Existing Test

You can open an existing test in order to enhance or run it.

To open an existing test:

1 Click the **Open** button or choose **File** > **Open**. The Open QuickTest Test dialog box opens.

2 Select a test and click **Open**. The test opens and the title bar displays the test name.

You can also open tests that are part of a TestDirector project. For more information, see Chapter 28, "Working with TestDirector."

Saving a Test

You can save a new test or save changes to an existing test.

To save a new test:

1 Click the **Save** button or choose **File** > **Save** to save the test. The Save QuickTest Test dialog box opens.

2 Choose the folder in which you want to save the test.

3 Type a name for the test in the **File** name box.

4 Click **Save**. QuickTest displays the test name in the title bar.

To save changes to an existing test:

➤ Click the **Save** button or choose **File** > **Save** to save changes to the test.

➤ Choose **File** > **Save As** to save an existing test to a new name or a new location.

You can also save tests to a TestDirector project. For more information, see Chapter 28, "Working with TestDirector."

Zipping a Test

While you record, QuickTest creates a series of configuration, data, and source code files. These files contain runtime and setup information. QuickTest saves these files together with the test. You can zip these files to conserve space and make the tests easier to transfer.

To zip a test:

1 Choose **File** > **Export to Zip File**. The Export to Zip File dialog box opens.

2 Type a zip file name and path, or accept the default name and path, and click **OK**. QuickTest zips the test and its associated files.

Unzipping a Test

To use a zipped test in QuickTest, you must use the Import from Zip File command to unzip it.

To unzip a zipped test:

1 Select **File** > **Import from Zip File**. The Import from Zip File dialog box opens.

2 Type or select the zip file that you want to unzip, choose a target folder into which you want to unzip the files, and click **OK**. QuickTest unzips the test and its associated files.

Printing a Test

You can print your test.

To print a test:

1 Click the **Print** button or choose **File** > **Print**. The Print dialog box opens.

2 Click **OK** to print.

6

Creating Checkpoints

You can check objects in your Web-based application to ensure that it functions as desired.

This chapter describes:

- Checking Objects
- Adding Checkpoints to a Test
- Understanding the Checkpoint Properties Dialog Box
- Modifying Checkpoints

About Creating Checkpoints

QuickTest enables you to add checks to your test. A *checkpoint* is a verification point that compares a current value for a specified property with the expected value for that property. This enables you to identify whether or not your Web site or application is functioning correctly.

When you add a checkpoint, QuickTest adds a checkpoint with an icon  under the highlighted step in the test tree. When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails. You can view the results of the checkpoint in the Test Results window.

Checking Objects

You can create checkpoints to check various objects in your Web-based application or application. You can create checkpoints for standard Windows, Web, ActiveX, Java, and multimedia objects.

- **Web objects:** Perform checks on Web page properties, text strings, tables, images and form elements. You can also use formulas to check that the data displayed on a Web page is valid. For more information, see Chapter 7, "Checking Web Objects."
- **ActiveX objects:** Perform checks on ActiveX controls in Microsoft Internet Explorer. For more information, see Chapter 30, "Testing ActiveX Controls."
- **Java objects:** Perform checks on Java objects. All standard Java objects from AWT, JFC, or Oracle toolkits are supported. For more information, see Chapter 31, "Testing Java Applets and Applications."
- **Multimedia objects:** Perform checks on Macromedia Flash objects and for Real Player application and controls. For more information about checking multimedia objects, see Chapter 32, "Testing Multimedia Applications."

Note: Java and multimedia objects are supported in QuickTest Professional only.

Adding Checkpoints to a Test

You can add checkpoints during or after recording a test. It is generally more convenient to define checks once the initial test has been recorded.

There are several ways to add checkpoints. To add checkpoints during or after recording:

- 
Use the commands on the Insert menu, or click the arrow beside the **Insert Checkpoint** button on the Main toolbar. This displays a menu of checkpoint options that are relevant to the selected step in the test tree.

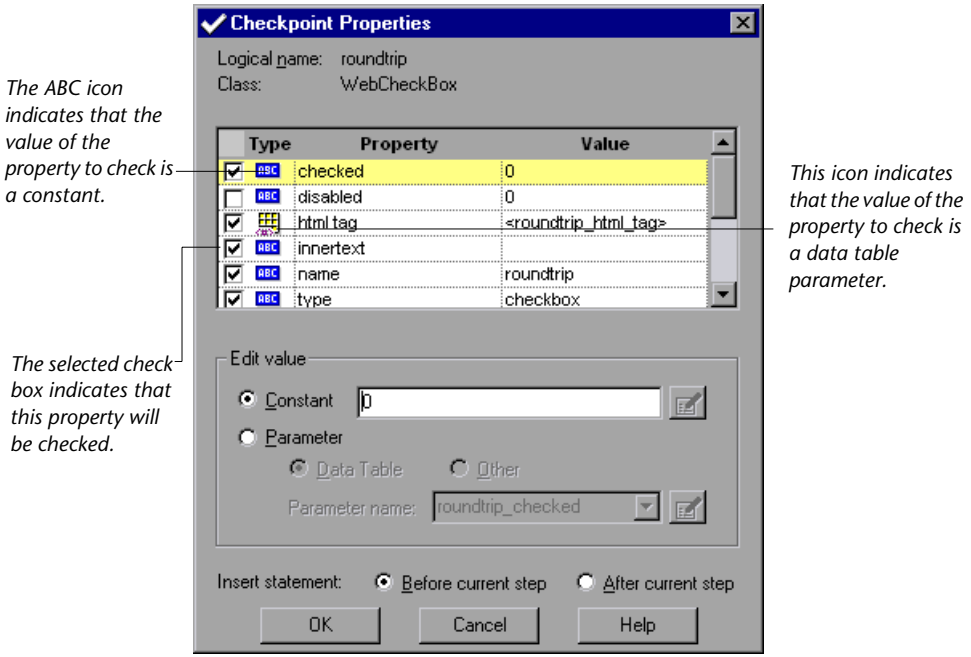
To add a checkpoint after recording only:

➤ Right-click the step where you want to add the checkpoint and choose **Insert Checkpoint**.

➤ Right-click in the ActiveScreen and choose **Insert Checkpoint**. This option can be used only after you record a test.

Understanding the Checkpoint Properties Dialog Box

While the Checkpoint Properties dialog box varies slightly depending on the type of object you are checking, the Checkpoint Properties dialog box generally includes the following basic elements:



Note: The Page Checkpoint Properties, Text Checkpoint Properties, Checkpoint Properties (for tables and data bases), Bitmap Checkpoint Properties, and Accessibility Checkpoint Properties dialog boxes are quite different from the Checkpoint Properties dialog box described below. For more information, see "Understanding the Page Checkpoint Properties Dialog Box," on page 73, "Understanding the Text Checkpoint Properties Dialog Box," on page 90, "Understanding the Checkpoint Properties Dialog Box," on page 115 (for tables), "Understanding the Checkpoint Properties Dialog Box," on page 129 (for databases), "Checking a Bitmap," on page 140, and "Creating Individual Accessibility Checkpoints," on page 87.

Identifying the Object

The top part of the dialog box displays basic information about the object to check such as the name and type of object.

Choosing which Property to Check

The next part of the dialog box displays the available properties for the object, and enables you to select which properties to check.

You can also create checkpoints on objects with variable descriptions. For more information, see Chapter 4, "Managing Test Objects."

Modifying the Expected Value

In the **Edit value** section, you can edit the value of any property that you want to check.

If the expected value of one of the properties you are checking changes, you don't have to rerecord the object. You can modify the expected value by selecting the relevant property and changing its value in the **Constant** box.

You can parameterize the expected value of an object by entering a parameter name in the **Parameter** box and entering the expected values in the appropriate column in the data table. For more information on parameterization, see Chapter 10, "Parameterizing Tests."

For more information on data tables, see Chapter 15, “Working with Data Tables.” For more information on regular expressions, see Chapter 12, “Using Regular Expressions.”

Modifying Checkpoints

If you already created a checkpoint, or if QuickTest automatically created page checkpoints, you can modify the settings. For example, you can choose to use parameters, or you can use filters to specify which image sources and links to check.

To modify a checkpoint:

1. Right-click an existing checkpoint in the test tree and choose **Checkpoint Properties**. A checkpoint dialog box opens.
2. Modify the properties and click **OK**.

7

Checking Web Objects

By adding Web object checkpoints to your tests, you can compare Web objects in different versions of your Web site.

This chapter describes:

- Recording and Running Tests on Web Sites
- Checking Pages
- Checking Web Content Accessibility
- Checking Text
- Checking Objects
- Checking Images
- Checking Tables

About Checking Web Objects

You can check the Web objects in your Web site to ensure that your Web site functions as desired. Web object checkpoints compare the expected values of object properties captured during the recording of the test to the object's current values during a test run. You can perform checks on Web page properties, text, tables, and other Web objects such as images and form elements.

Recording and Running Tests on Web Sites

You record on Web browsers to create tests to check Web objects. QuickTest supports recording and running tests on the following Web browsers:

- Netscape Navigator
- Microsoft Internet Explorer
- AOL (America Online)

QuickTest tests are cross-browser: you can record a test on one browser and run it on any other browser.

Notes about recording tests on the AOL browser:

QuickTest supports recording on the Back, Forward, Home, and Reload buttons. It records the address bar and the Go button using the **Navigate** method.

To record and run tests on the AOL browser, you must click **Use a Web browser or another application that is open** in the Browser Settings dialog box.

For additional information on working with the AOL browser, refer to the *ReadMe* file.

For information on supported browser versions, refer to the *ReadMe* file.

Checking Pages

You can check statistical information about your Web pages by adding page checkpoints to your test. These checkpoints check the links and the source of images on a Web page. You can also instruct page checkpoints to include a check for broken links.

Automatic Page Checkpoints

You can instruct QuickTest to create automatic page checkpoints for every page in all tests by selecting the **Create a checkpoint for each Web page while recording** check box in the Advanced Web Options dialog box (click the Advanced button in the Web tab of the Options dialog box). By default, the automatic page checkpoint includes the checks that you select from among the available options in the Advanced Web Options dialog box.

You can also instruct QuickTest not to perform automatic page checkpoints when you run your test by selecting the **Ignore automatic checkpoints while running tests** check box in the Advanced Web Options dialog box of the Web tab of the Options dialog box.

For more information, see Chapter 24, "Setting Global Testing Options."

Creating Individual Page Checkpoints

If you did not choose to add page checkpoints automatically while recording, you can add a page checkpoint to your test to check the links and the image sources on a selected Web page either while recording or after recording.

Note: You cannot create a new page checkpoint for a page if one has already been created automatically. You can modify a page checkpoint that has already been created, however, as described in "Understanding the Page Checkpoint Properties Dialog Box" on page 73.

To add a page checkpoint while recording:

1 Navigate to a page where you want to add a checkpoint.

2 Choose **Insert** > **Checkpoint** > **Standard Checkpoint**, or click the **Insert Checkpoint** button and click in the page you want to check. The Object Selection - Checkpoint Properties dialog box opens.

3 Select the **Page** item and click **OK**. The Page Checkpoint Properties dialog box opens.

4 Modify the settings for the checkpoint in the Page Checkpoint Properties dialog box, as described in "Understanding the Page Checkpoint Properties Dialog Box" on page 73.

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

A tree item with a checkpoint icon is added to your test tree.

To add a page checkpoint after recording:

1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to add a checkpoint.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click anywhere on the ActiveScreen and choose **Insert Checkpoint.** The Object Selection - Checkpoint Properties dialog box opens.

4 Select the **Page** item you want to check from the displayed object tree. Click **OK**. The Page Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "Understanding the Page Checkpoint Properties Dialog Box" on page 73.

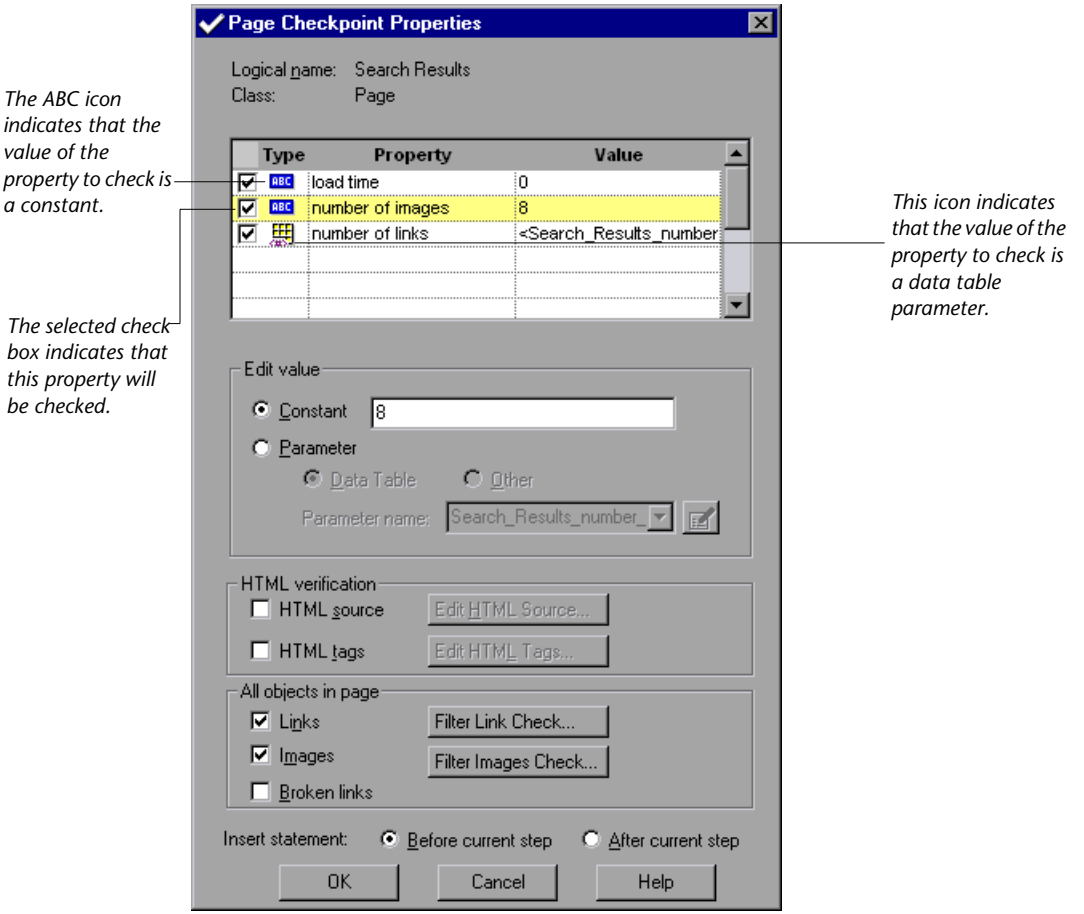
6 Click **OK** to close the dialog box.

A tree item with a checkpoint icon is added to your test tree.

Note: You cannot select the **HTML Verification** options while creating a page checkpoint from the ActiveScreen. You can select these options only when creating a page checkpoint while recording.

Understanding the Page Checkpoint Properties Dialog Box

The Page Checkpoint Properties dialog box enables you to choose which properties to check.



Identifying the Object

The top part of the dialog box displays information about the object to check:

Information	Description
Logical name	The title of the Web page as defined in the HTML code.
Class	The type of object. This is always Page.

Choosing which Property to Check

The default properties for the object are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

Pane Element	Description
check box	For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently a data table parameter. The  icon indicates that the value of the property is currently an environment variable parameter. The  icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The value of the property. Note that unless you edit this value, the value in the page will be the expected value of the property when you run your test. For information about editing the value of a property, see "Editing the Value of a Page Property," on page 75.

Editing the Value of a Page Property

In the Edit value section, you use the following options to edit the value of the property to check:

Option	Description
Constant (default)	Sets the expected value of the property as a constant. Click the **Edit Constant Value Options** button  to open the Constant Value Options dialog box, where you can specify the value as a text string or a regular expression. The value of the property is constant for each iteration of the test run.
Parameter	Sets the expected value of the property as a parameter. For more information, see Chapter 10, “Parameterizing Tests.”
Data Table	Specifies the parameter as a Data Table parameter. The value of the property is determined by the data in the Data pane. For more information about creating Data Table parameters, see “Data Table Parameters,” on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.

Option	Description
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can set the value as a regular expression. You can also specify the parameter name and set the parameter as global or current action data. For more information, see Chapter 14, "Working with Actions." If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

Checking the HTML Source

In the HTML Verification section, you can use the following options to check the HTML source and tags of the page:

Option	Description
HTML Source	Checks that the source in the web page being tested matches the expected HTML code (the source code of the page at the time that the test is recorded).
Edit HTML Source (enabled only when the HTML Source check box is selected)	Opens the HTML Source dialog box, which displays the expected HTML code. Note that you can also use regular expressions when editing the expected HTML source code if you click the regular expression check box at the bottom of the page. Edit the expected HTML source code and click **OK**.

Option	Description
HTML Tags	Checks that the HTML tags in the web page being tested match the expected HTML tags (the HTML tags on the page at the time that the test is recorded).
Edit HTML Tags (enabled only when the HTML Tags check box is selected)	Opens the HTML Tags dialog box, which displays the expected HTML tags. Note that you can also use regular expressions when editing the HTML tags if you click the regular expression check box at the bottom of the page. Edit the expected HTML tags and click **OK**.

Note: The **HTML Verification** options are available only when creating a page checkpoint while recording. They are not available when creating a page checkpoint from the ActiveScreen.

Checking All the Objects in a Page

In the All objects in page section, you can check all the links, images, and broken links in a page. You can use the following options to check the objects in a page:

Option	Description
Links	Checks the functionality of the links in the page according to your selections in the Filter Link Checks dialog box.
Filter Link Check (enabled only when the Links check box is selected)	Opens the Filter Link Checks dialog box, which enables you to specify which hypertext links to check in the page. For additional information, see “Filtering Hypertext Links” on page 79.
Images	Checks that the images are displayed on the page according to your selections in the Filter Image Check dialog box.

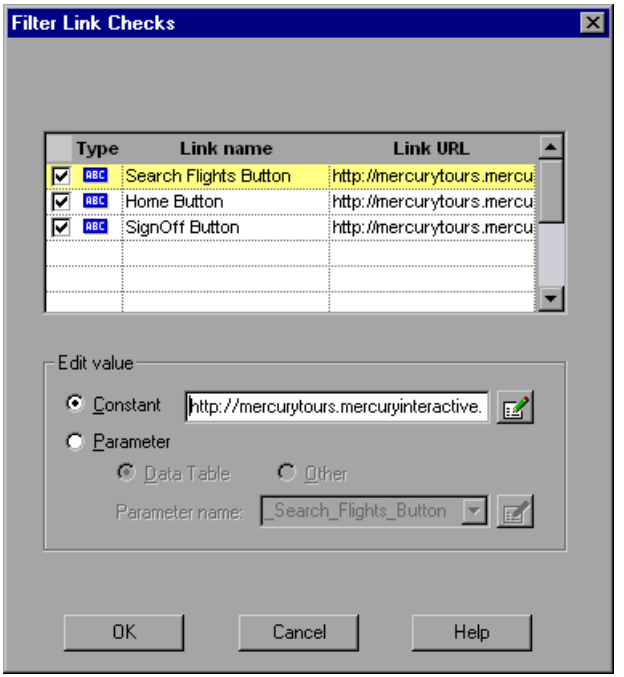
Option	Description
Filter Images Check (enabled only when the Images check box is selected)	Opens the Filter Image Check dialog box, which enables you to specify which image sources to check in the page. For additional information, see "Filtering Image Sources" on page 83.
Broken Links	Checks for broken links in the page.

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to check the value of the page property before the highlighted step is performed. Choose **After current step** if you want to check the value of the page property after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding a page checkpoint to an existing test. It is disabled when recording a new test or modifying an existing page checkpoint.

Filtering Hypertext Links

You can filter which hypertext links to check in a page checkpoint using the Filter Link Checks dialog box. You open this dialog box from the Page Checkpoint Properties dialog box. For additional information, see "Checking All the Objects in a Page" on page 77.



Choosing which Hypertext Links to Check

You can use the following options to choose which hypertext links to check in a page checkpoint:

Pane Element	Description
check box	Each link on the page has a corresponding check box. To check a link, select the corresponding check box (by default all links are selected). To exclude a link from the page checkpoint, clear the corresponding check box.
Type	The  icon indicates that the target URL is currently a constant. The  icon indicates that the target URL is currently a data table parameter. The  icon indicates that the target URL is currently an environment variable parameter. The  icon indicates that the target URL is currently a random number parameter.
Link name	The text in the hypertext link.
Link URL	The target URL.

Editing the Value of the Target URL

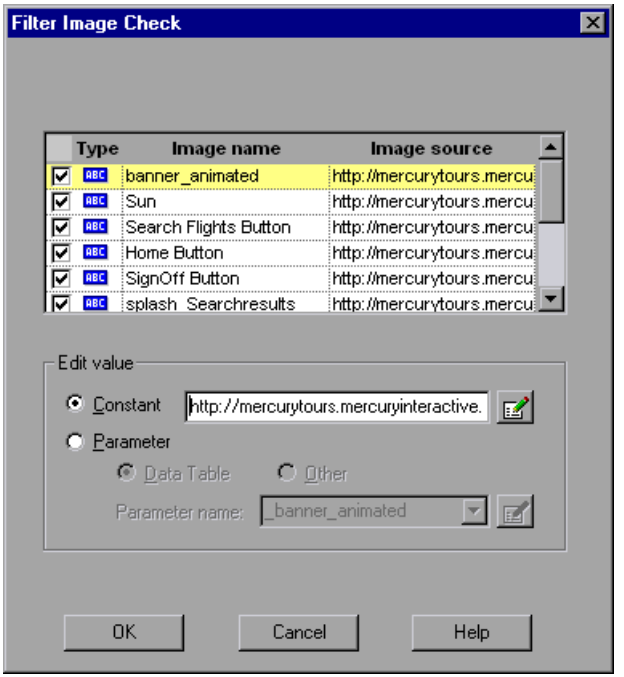
In the Edit Value section, you use the following options to edit the value of the target URL to which the hypertext links:

Option	Description
Constant (default)	Sets the expected value of the target URL as a constant. Click the **Edit Constant Value Options** button  to open the Constant Value Options dialog box, where you can specify the value as a text string or a regular expression. The value of the target URL is constant for each iteration of the test run.
Parameter	Sets the expected value of the target URL as a parameter. For more information, see Chapter 10, “Parameterizing Tests.”
Use data table formula (advanced)	Sets a data table containing a formula as a parameter. For more information, see Chapter 10, “Parameterizing Tests.”
Data Table	Specifies the parameter as a Data Table parameter. The value of the target URL is determined by the data in the Data pane. For more information about creating Data Table parameters, see “Data Table Parameters,” on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.

Option	Description
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can set the value as a regular expression. You can also specify the parameter name and set the parameter as global or current action data. For more information, see Chapter 14, "Working with Actions." If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

Filtering Image Sources

You can filter which image sources to check in a page checkpoint using the Filter Image Check dialog box. You open this dialog box from the Page Checkpoint Properties dialog box. For additional information, see "Checking All the Objects in a Page" on page 77.



Choosing which Image Sources to Check

You can use the following options to choose which image sources to check in a page checkpoint:

Pane Element	Description
check box	Each image source on the page has a corresponding check box. To check an image source, select the corresponding check box (by default all image sources are selected). To exclude an image source from the page checkpoint, clear the corresponding check box.
Type	The  icon indicates that the image source is currently a constant. The  icon indicates that the image source is currently a data table parameter. The  icon indicates that the image source is currently an environment variable parameter. The  icon indicates that the image source is currently a random number parameter.
Image name	The name of the image.
Image source	The image source file and path.

Editing the Value of the Path of the Image Source File

In the Edit Value section, you use the following options to edit the path of the image source file:

Option	Description
Constant (default)	Sets the expected value of the path of the image source file as a constant. Click the **Edit Constant Value Options** button  to open the Constant Value Options dialog box, where you can specify the value as a text string or a regular expression. The value of the path of the image source file is constant for each iteration of the test run.
Parameter	Sets the expected value of the path of the image source file as a parameter. For more information, see Chapter 10, “Parameterizing Tests.”
Data Table	Specifies the parameter as a Data Table parameter. The value of the path of the image source file is determined by the data in the Data pane. For more information about creating Data Table parameters, see “Data Table Parameters,” on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.

Option	Description
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can set the value as a regular expression. You can also specify the parameter name and set the parameter as global or current action data. For more information, see Chapter 14, "Working with Actions." If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

Checking Web Content Accessibility

You can add accessibility checkpoints to help you quickly identify areas of your Web site that may not conform to the W3C Web Content Accessibility Guidelines. You can add automatic accessibility checkpoints to each page in your test, or you can add individual accessibility checkpoints to individual pages or frames.

Setting Accessibility Checkpoint Preferences

All accessibility checkpoints in your test use the options selected in the Advanced Web Options dialog box at the time of the test run. For information about the accessibility checkpoint options, see "Advanced Web Options," on page 399.

Automatic Accessibility Checkpoints

You can instruct QuickTest to create automatic accessibility checkpoints for every page in all tests by selecting the **Add Automatic accessibility checkpoint to each Web page while recording** check box in the Advanced Web Options dialog box (click the Advanced button in the Web tab of the Options dialog box). If you select this option, an accessibility checkpoint is inserted for each page as you record.

Creating Individual Accessibility Checkpoints

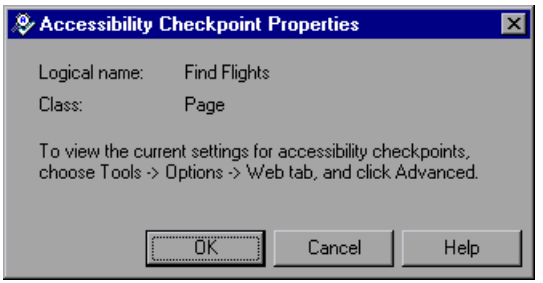
If you did not choose to add accessibility checkpoints automatically while recording, you can add an accessibility checkpoint to help you quickly identify areas of a particular Web page or frame that may not conform to the W3C Web Content Accessibility Guidelines. You can add accessibility checkpoints either while recording or after recording.

To add an accessibility checkpoint while recording:

1 Navigate to a page where you want to add an accessibility checkpoint.

2 Choose **Insert** > **Checkpoint** > **Accessibility Checkpoint**, or click the **Insert Checkpoint** button and click in the page or frame you want to check.

- If the page contains frames, the Object Selection - Accessibility Checkpoint Properties dialog box opens. Select the **Page** or **Frame** item, and click **OK**. The Accessibility Checkpoint Properties dialog box opens.
- If the page does not contain frames, the Accessibility Checkpoint Properties dialog box opens.



Note: If you check the frame, the class in the Accessibility Checkpoint Properties dialog box is Frame.

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

A tree item with a checkpoint icon is added to your test tree.

To add an accessibility checkpoint after recording:

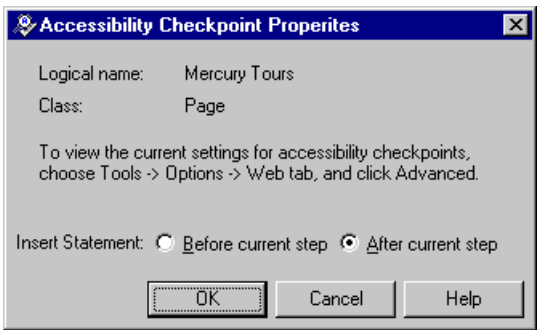
1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to add a checkpoint.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click anywhere on the ActiveScreen, and choose **Insert Accessibility Checkpoint.**

- If the page contains frames, the Object Selection - Accessibility Checkpoint Properties dialog box opens. Select the **Page** or **Frame** item, and click **OK**. The Accessibility Checkpoint Properties dialog box opens.
- If the page does not contain frames, the Accessibility Checkpoint Properties dialog box opens.



Note: If you check the frame, the class in the Accessibility Checkpoint Properties dialog box is Frame.

Choose **Before current step** if you want to check the accessibility elements before the highlighted step is performed. Choose **After current step** if you want to check the accessibility elements after the highlighted step is performed.

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

A tree item with a checkpoint icon is added to your test tree.

Checking Text

You can check that a specified text string is displayed on your Web page by adding a text checkpoint to your test. To add a text checkpoint to your test, you use the Text Checkpoint Properties dialog box.

Creating a Text Checkpoint

You can add a text checkpoint while recording or afterward.

To add a text checkpoint while recording:

1 Highlight a text string on the Web page.

2 Choose **Insert** > **Checkpoint** > **Text Checkpoint**, or click the arrow next to the Insert Checkpoint button, and choose **Text Checkpoint**.

The QuickTest window is minimized, and the mouse pointer turns into a pointing hand.

3 Click the text string. The Text Checkpoint Properties dialog box opens.

4 Specify the settings for the checkpoint. For more information, see “Understanding the Text Checkpoint Properties Dialog Box,” on page 90.

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

A tree item with a checkpoint icon is added to your test tree.

To add a text checkpoint after recording:

1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to add a checkpoint.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Highlight a text string on the ActiveScreen.

4 Right-click the text string and choose **Insert Text Checkpoint**. The Text Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "Understanding the Text Checkpoint Properties Dialog Box," on page 90.

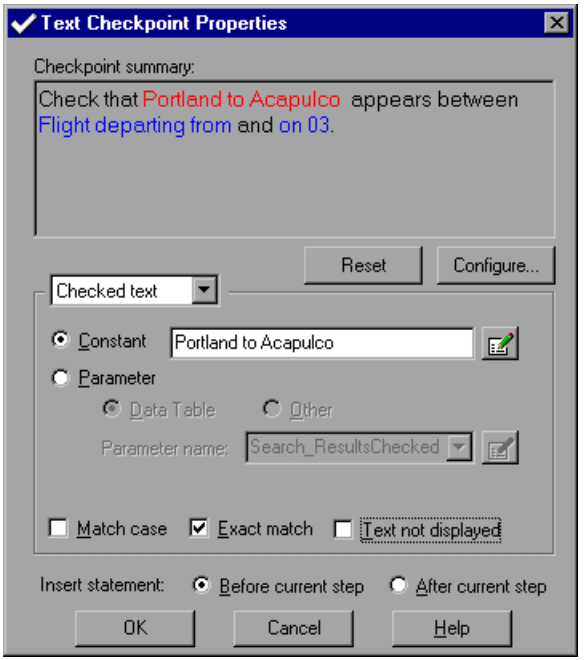
6 Click **OK** to close the dialog box.

A tree item with a checkpoint icon is added to your test tree.

Understanding the Text Checkpoint Properties Dialog Box

In the Text Checkpoint Properties dialog box, you can specify the checked text as well as which text is displayed before and after the checked text. This is particularly helpful when the text string you want to check is displayed several times in the same Web page. For example, you want to check a text string that is displayed several times in a Web page. Suppose you want to check only the third occurrence of the text string. To check for the text string in a specific location, you can specify which text precedes and/or

follows the text string you are checking and to which occurrence of the specified text strings you are referring.



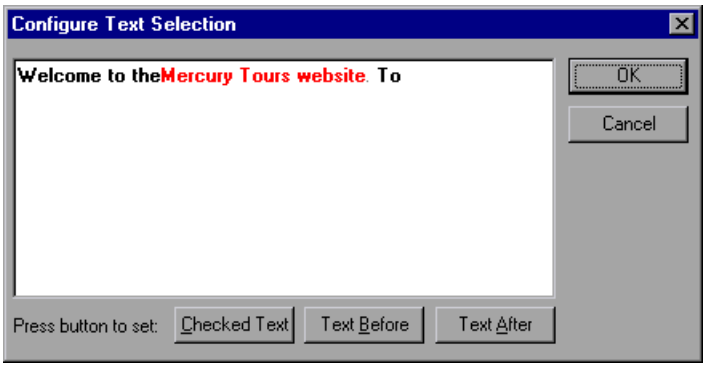
The Checkpoint summary pane at the top of the dialog box summarizes the selected text for the checkpoint. It displays the text you highlighted in the Web page when creating the text checkpoint, plus some text before and after the highlighted text. QuickTest displays the checked text in red and the text before and after the checked text in blue.

Configuring the Text Selection

You can configure the text selection displayed in the Checkpoint summary pane using the following options:

Option	Description
Configure	Opens the Configure Text Selection dialog box, where you can specify the checked text, the text before (if any), and the text after (if any). **Note:** If you modifying the configuration of an existing text checkpoint, you must redefine the other checkpoint settings.
Checkpoint summary	Displays the text configuration you make using the Configure Text Selection dialog box.
Reset	Resets the text selection to the previous configuration.

The Configure Text Selection dialog box displays the text you highlighted in the Web page when creating the text checkpoint, plus some text before and after the highlighted text. QuickTest displays the checked text in red and the text before and after in black.



Note: If you are modifying an existing text checkpoint, the Configure Text Selection dialog box displays the previous text configuration.

Tip: If you want to configure more text than is displayed, cancel the text checkpoint and make a larger selection in your Web page.

You can specify the parts of the displayed text for the checkpoint by highlighting them and using the following options:

Option	Description
Checked Text	Sets the highlighted text as the checked text. QuickTest displays this text in red and the remainder in black.
Text Before	Sets the highlighted text as the text before the checked text.
Text After	Sets the highlighted text as the text after the checked text.

Note: QuickTest displays in grey any text that is not configured as checked text, text before, or text after. The grey text is not displayed the next time the Configure Text Selection dialog box is opened.

When you close the Configure Text Selection dialog box, the Checkpoint summary pane displays the new text selection configuration.

Specifying the Checked Text

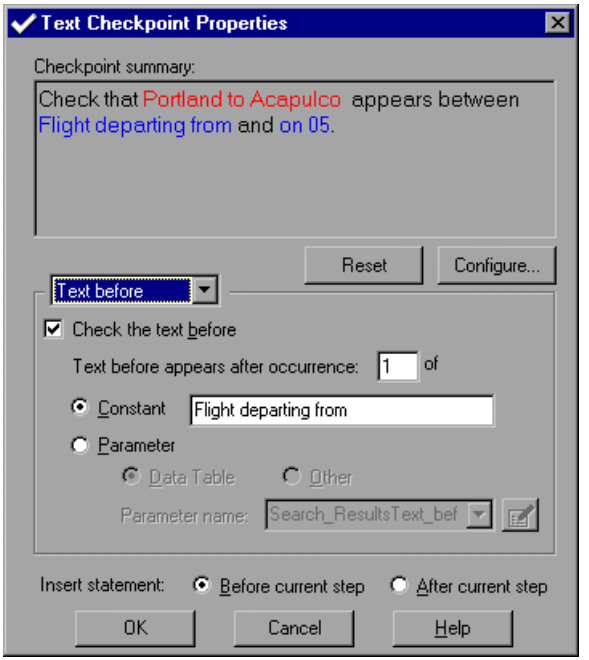
Choose **Checked Text** from the list box. In the Checked Text section, you can specify the checked text as a constant or a parameter. QuickTest displays the following options for specifying the checked text:

Option	Description
Constant (default)	Sets the expected value of the checked text as a constant. Click the **Edit Constant Value Options** button  to open the Constant Value Options dialog box, where you can specify a text string or a regular expression. The value of the selected text is constant for each iteration of the test run. **Tip:** The text box displays the checked text. You can change the checked text by typing in the text box, or by using the Configure Text Selection dialog box.
Parameter	Sets the expected value of the checked text as a parameter. For more information, see Chapter 10, "Parameterizing Tests."
Data Table	Specifies the parameter as a Data Table parameter. The value of the checked text is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.

Option	Description
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can set the value as a regular expression, or choose to use a data table formula. You can also specify the parameter name and set the parameter as global or current action data. For more information, see Chapter 14, "Working with Actions." If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.
Match case	Conducts a case sensitive check.
Exact match	Checks according to the exact expected text. For example, you create a checkpoint with the following description: *Check that New York is displayed between Flight departing from and to San Francisco* and select Exact match. If the actual text is *New York City*, the checkpoint fails. If you do not select Exact match, the checkpoint passes because the expected text is contained within the actual text.
Text not displayed	Checks that the text string is not displayed. For example, you create a checkpoint with the following description: *Check that New York is displayed between Flight departing from and to San Francisco* and select Text not displayed. QuickTest checks that the text *New York* is not displayed.

Specifying the Text Displayed Before the Checked Text

Choose **Text Before** from the list box. In the Text Before section, you can specify the text before the checked text as a constant or a parameter.



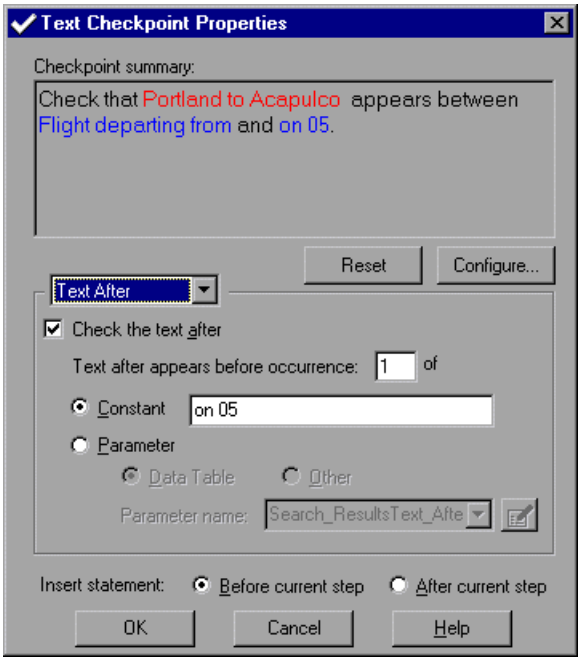
QuickTest displays the following options for specifying which text is displayed before the checked text:

Option	Description
Check the text before	Checks the text before the checked text. To ignore this text, clear this check box.
Text to check is displayed after occurrence	Checks that the checked text is displayed after the specified text. If the identical text string you specify is displayed more than once on the page, you can specify to which occurrence of the string you are referring. If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate. If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words "Mercury Tours" are displayed after the fourth occurrence of the word "the", enter 4 in the **Text to check is displayed after occurrence** box.
Constant (default)	Sets the expected value of the text before the checked text as a constant. The value is constant for each iteration of the test run. Note that the text box displays the text before the checked text. You can change the text by typing in the text box, or by using the Configure Text Selection dialog box. **Tip:** If you modify the text, use a unique string whenever possible so that the occurrence number is 1.
Parameter	Sets the expected value of the text before the checked text as a parameter. For more information, see Chapter 10, “Parameterizing Tests.”
Data Table	Specifies the parameter as a Data Table parameter. The value of the text before the checked text is determined by the data in the Data pane. For more information about creating Data Table parameters, see “Data Table Parameters,” on page 152.

Option	Description
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can also specify the parameter name and set the parameter as global or current action data. If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

Specifying the Text Displayed After the Checked Text

Choose **Text After** from the list box. In the Text After section, you can specify the text after the checked text as a constant or a parameter.



QuickTest displays the following options for specifying which text is displayed after the checked text:

Option	Description
Check the text after	Checks the text after the checked text. To ignore this text, clear this check box.
Text to check is displayed before occurrence	Checks that the checked text is displayed before the specified text. QuickTest starts counting occurrences of the **is displayed before** text you specify, from the end of the **is displayed after** string. In other words, it starts looking for the specified text from the text you selected to check. If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the recommended text string and the string you specify is displayed in your highlighted text as well as in the **is displayed before** text, you need to modify the occurrence number accordingly. For example, if you want to check that the words "my hat is the best" are displayed before the word "hat", enter 2 in the **Text to check is displayed before occurrence** box, to show that you want your text to be displayed before the second occurrence of the word "hat".
Constant (default)	Sets the expected value of the text after the checked text as a constant. The value is constant for each iteration of the test run. Note that the text box displays the text after the checked text. You can change the text by typing in the text box, or by using the Configure Text Selection dialog box. **Tip:** If you modify the text, use a unique string whenever possible so that the occurrence number is always 1.
Parameter	Sets the expected value of the text after the checked text as a parameter. For more information, see Chapter 10, "Parameterizing Tests."

Option	Description
Data Table	Specifies the parameter as a Data Table parameter. The value of the text after the checked text is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can also specify the parameter name and set the parameter as global or current action data. If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to check the value of the text before the highlighted step is performed. Choose **After current step** if you want to check the value of the text after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding a text checkpoint to an existing test. It is disabled when recording a new test or modifying an existing text checkpoint.

Checking Objects

You can check that a specified object is displayed on your Web page by adding an object checkpoint to your test. To add an object checkpoint to your test, you use the Checkpoint Properties dialog box.

Creating an Object Checkpoint

You can add an object checkpoint while recording or afterward.

To add an object checkpoint while recording:

1 Click the **Insert Checkpoint** button or choose **Insert** > **Checkpoint** > **Standard Checkpoint**.

The QuickTest window is minimized, and the mouse pointer turns into a pointing hand.

2 Click the object you want to check. The Object Selection - Checkpoint Properties dialog box opens.

3 Select the item you want to check from the displayed object tree. The tree item name depends on the object's class, for example:

icon	Object	Class
	Check box	WebCheckBox
	Edit box	WebEdit
	Radio button	WebRadioGroup
	List box	WebList
	Element	WebElement

4 Click **OK**. The Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box," on page 104.

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

A tree item with a checkpoint  icon is added to your test tree.

To add an object checkpoint after recording:

1 Make sure the **Display Views** button is selected.

Click a step in your test where you want to add a checkpoint.

The ActiveScreen displays the Web page corresponding to the highlighted step.

2 Right-click an object on the ActiveScreen and choose **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.

3 Select the item you want to check from the displayed object tree. The tree item name depends on the object's class, for example:

icon	Object	Class
	Check box	WebCheckBox
	Edit box	WebEdit
	Radio button	WebRadioGroup
	List box	WebList
	Element	WebElement

4 Click **OK**. The Checkpoint Properties dialog box opens.

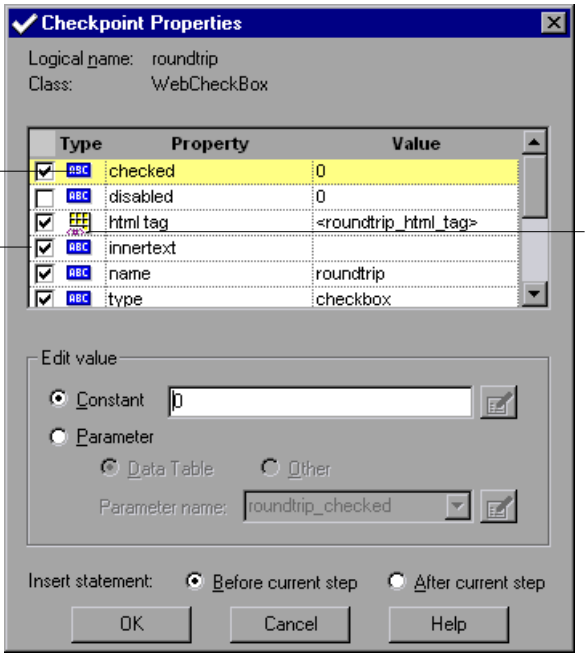
5 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box," on page 104.

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

A tree item with a checkpoint icon is added to your test tree.

Understanding the Checkpoint Properties Dialog Box

In the Checkpoint Properties dialog box, you can specify which properties of the object to check, and edit the values of these properties.



The ABC icon indicates that the value of the property to check is a constant.

The selected check box indicates that this property will be checked.

This icon indicates that the value of the property to check is a data table parameter.

Identifying the Object

The top part of the dialog box displays information about the object to check:

Information	Description
Logical name	The name of the object as defined in the HTML code of the Web page.
Class	The type of object. In this example, the "WebCheckBox" class indicates that the object is a check box.

Choosing which Property to Check

The default properties for the object are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

Pane Element	Description
check box	For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently a data table parameter. The  icon indicates that the value of the property is currently an environment variable parameter. The  icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property to check.
Value	The value of the property. Note that unless you edit this value, the listed value will be the expected value of the property when you run your test. For information about editing the value of a property, see “Editing the Value of an Object Property,” on page 106.

Editing the Value of an Object Property

In the Edit Value section, you use the following options to edit the value of the property to check:

Option	Description
Constant (default)	Sets the expected value of the property as a constant. Click the **Edit Constant Value Options** button  to open the Constant Value Options dialog box, where you can specify a text string or a regular expression. The value of the property is constant for each iteration of the test run.
Parameter	Sets the expected value of the property as a parameter. For more information, see Chapter 10, "Parameterizing Tests."
Data Table	Specifies the parameter as a Data Table parameter. The value of the property is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.

Option	Description
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can set the value as a regular expression, or choose to use a data table formula. You can also specify the parameter name and set the parameter as global or current action data. For more information, see Chapter 14, "Working with Actions." If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to check the value of the object property before the highlighted step is performed. Choose **After current step** if you want to check the value of the object property after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding an object checkpoint to an existing test. It is disabled when recording a new test or modifying an existing object checkpoint.

Checking Images

You can check that a specified image is displayed on your Web page by adding an image checkpoint to your test. To add an image checkpoint to your test, you use the Image Checkpoint Properties dialog box.

Creating an Image Checkpoint

You can add an image checkpoint while recording or afterward.

To add an image checkpoint while recording:

1 Click the **Insert Checkpoint** button or choose **Insert** > **Checkpoint** > **Standard Checkpoint**.

The QuickTest window is minimized, and the mouse pointer turns into a pointing hand.

2 Click the image you want to check. The Object Selection - Checkpoint Properties dialog box opens.

3 Select the **Image** item you want to check from the displayed object tree.

4 Click **OK**. The Image Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "Understanding the Image Checkpoint Properties Dialog Box," on page 109.

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

A tree item with a checkpoint icon is added to your test tree.

To add an image checkpoint after recording:

1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to add a checkpoint.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click an image on the ActiveScreen and choose **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.

4 Select the **Image** item you want to check from the displayed object tree.

5 Click **OK**. The Image Checkpoint Properties dialog box opens.

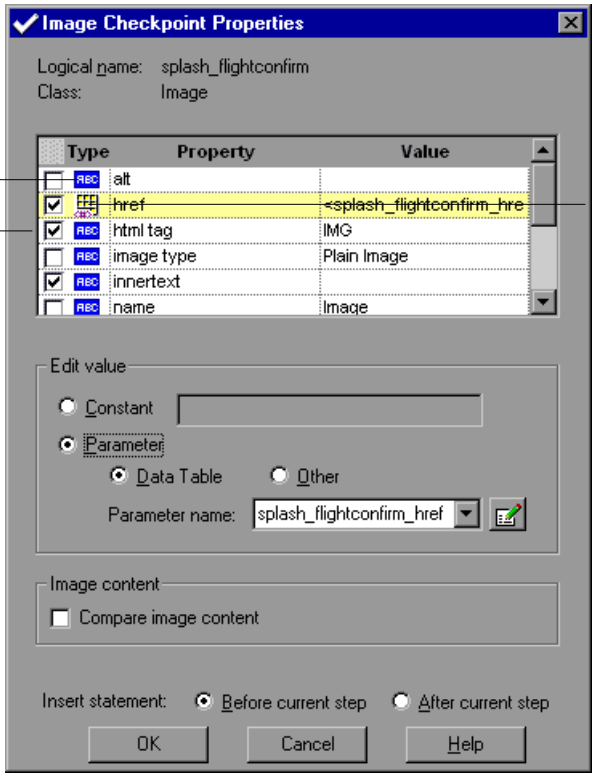
6 Specify the settings for the checkpoint. For more information, see "Understanding the Image Checkpoint Properties Dialog Box," on page 109.

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

A tree item with a checkpoint icon is added to your test tree.

Understanding the Image Checkpoint Properties Dialog Box

In the Image Checkpoint Properties dialog box, you can specify which properties of the image to check, and edit the values of these properties.



The ABC icon indicates that the value of the property to check is a constant.

The selected check box indicates that this property will be checked.

This icon indicates that the value of the property to check is a data table parameter.

Identifying the Image

The top part of the dialog box displays information about the image to check:

Information	Description
Logical name	The name of the image as defined in the HTML code of the Web page.
Class	The type of object. This is always image.

Choosing which Property to Check

The default properties for the image are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

Pane Element	Description
check box	For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently a data table parameter. The  icon indicates that the value of the property is currently an environment variable parameter. The  icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The value of the property. Note that unless you edit this value, the listed value will be the expected value of the property when you run your test. For information about editing the value of a property, see "Editing the Value of an Image Property," on page 111.

Editing the Value of an Image Property

In the Edit Value section, you use the following options to edit the value of the property to check:

Option	Description
Constant (default)	Sets the expected value of the property as a constant. The value of the property is constant for each iteration of the test run.
Parameter	Sets the expected value of the property as a parameter. For more information, see Chapter 10, "Parameterizing Tests."
Data Table	Specifies the parameter as a Data Table parameter. The value of the property is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.
Parameter Name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.

Option	Description
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box where you can choose to use a data table formula. You can also specify the parameter name and set the parameter as global or current action data. If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box where you can specify the parameter type and preferences.
Compare image content	Compares the graphic of the expected image source file with the graphic of the actual image source file in the Test Results. If they are identical, only one graphic is displayed.

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to check the value of the image property before the highlighted step is performed. Choose **After current step** if you want to check the value of the image property after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding an image checkpoint to an existing test. It is disabled when recording a new test or modifying an existing image checkpoint.

Checking Tables

You can check that a specified text string displays in a cell in a table on your Web page by adding a table checkpoint to your test. To add a table checkpoint to your test, you use the Checkpoint Properties dialog box.

Note that you can also create checkpoints on ActiveX controls and Java applets and applications that are tables using the information below. For additional information on testing ActiveX controls, see Chapter 30, "Testing ActiveX Controls." For additional information on testing Java applets and applications, see Chapter 31, "Testing Java Applets and Applications."

Note: Editing and saving a table checkpoint created in a previous version of QuickTest will convert the checkpoint to an updated format. Converting a checkpoint to the new format may modify some of the settings, so check that the settings are correct before saving the checkpoint. To keep the table checkpoint in its original format, click **Cancel** in the Checkpoint Properties dialog box.

Creating a Table Checkpoint

You can add a table checkpoint while recording or afterward.

To add a table checkpoint while recording:

1 Choose **Insert** > **Checkpoint** > **Standard Checkpoint** or click the **Insert Checkpoint** button.

The QuickTest window is minimized, and the mouse pointer turns into a pointing hand.

2 Click a table you want to check. The Object Selection - Checkpoint Properties dialog box opens.

3 Select a **WebTable** item from the displayed object tree.

4 Click **OK**. The Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box," on page 115.

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

A tree item with a checkpoint icon is added to your test file tree.

To add a table checkpoint after recording:

1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to add a checkpoint.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click the table on the ActiveScreen and choose **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.

4 Select a **WebTable** item from the displayed object tree.

5 Click **OK**. The Checkpoint Properties dialog box opens.

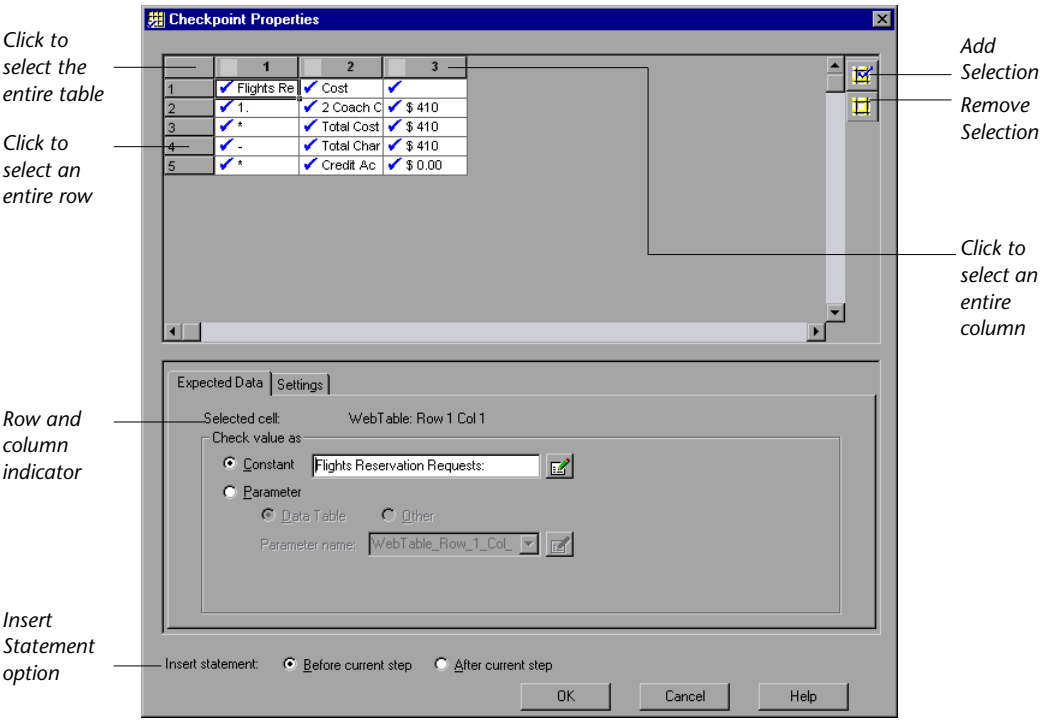
6 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box," on page 115.

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

A tree item with a checkpoint icon is added to your test file tree.

Understanding the Checkpoint Properties Dialog Box

In the Checkpoint Properties dialog box, you can choose which cells in the table to check and specify the settings for the table checkpoint.



Specifying which Cells to Check

The top part of the Checkpoint Properties dialog box displays a grid representing the cells in the table. The grid displays rows and columns of a table. You can check the entire table, a row, a column, or a cell. QuickTest only checks cells containing a blue check mark. By default, all cells contain a check mark.

To check/remove from check:	Double-click
a single cell. Note that in the **Check Value as** section of the Expected Data tab, the cell value is displayed in the **Constant** box.	the cell
an entire row	the row header
an entire column	the column header
the entire table	the top-left corner of the grid

Alternatively, you can click the **Add Selection** button to check an entire selection. To remove an entire selection from the check, you can click the **Remove Selection** button.

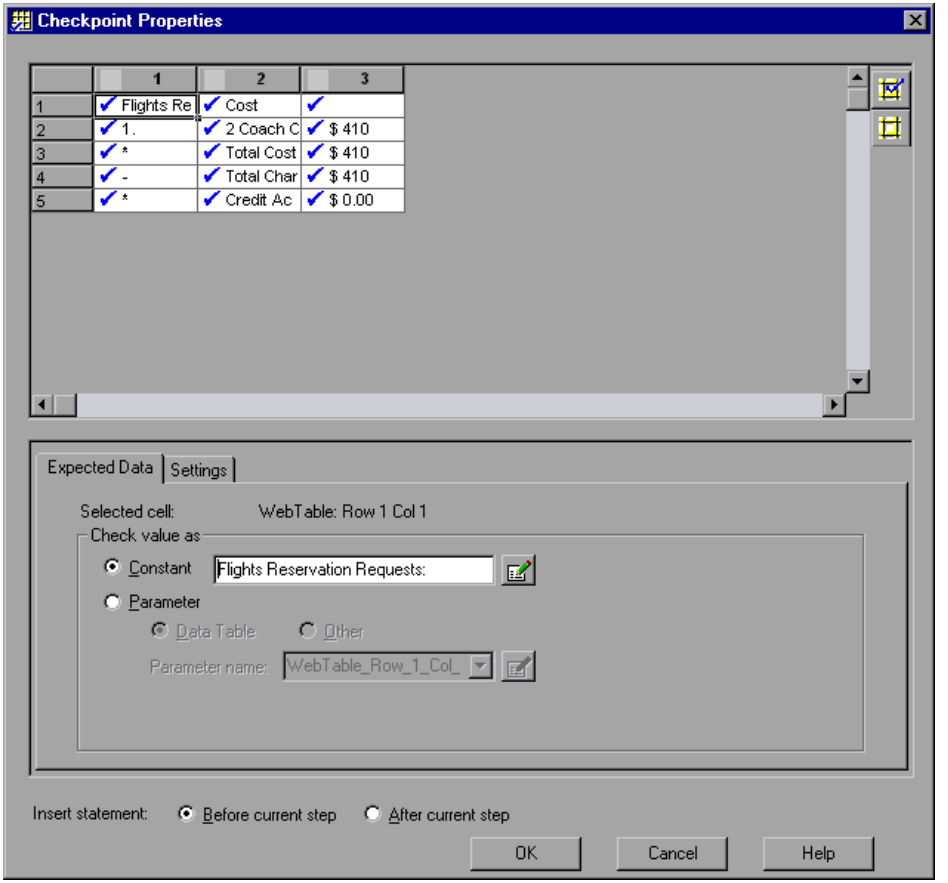
Note: When more than one cell is selected, the options on the Expected Data tab are disabled.

Double-clicking on the grid toggles the settings of each cell in the selection.

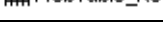
Tip: You can change the column widths and row heights of the grid by dragging the boundaries of the column and row headers.

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the highlighted cell in the table. QuickTest compares this value with the actual value that it finds in the table on your Web page. If the expected value and the actual value do not match, the test fails. You can determine the value of the cell using the following features:



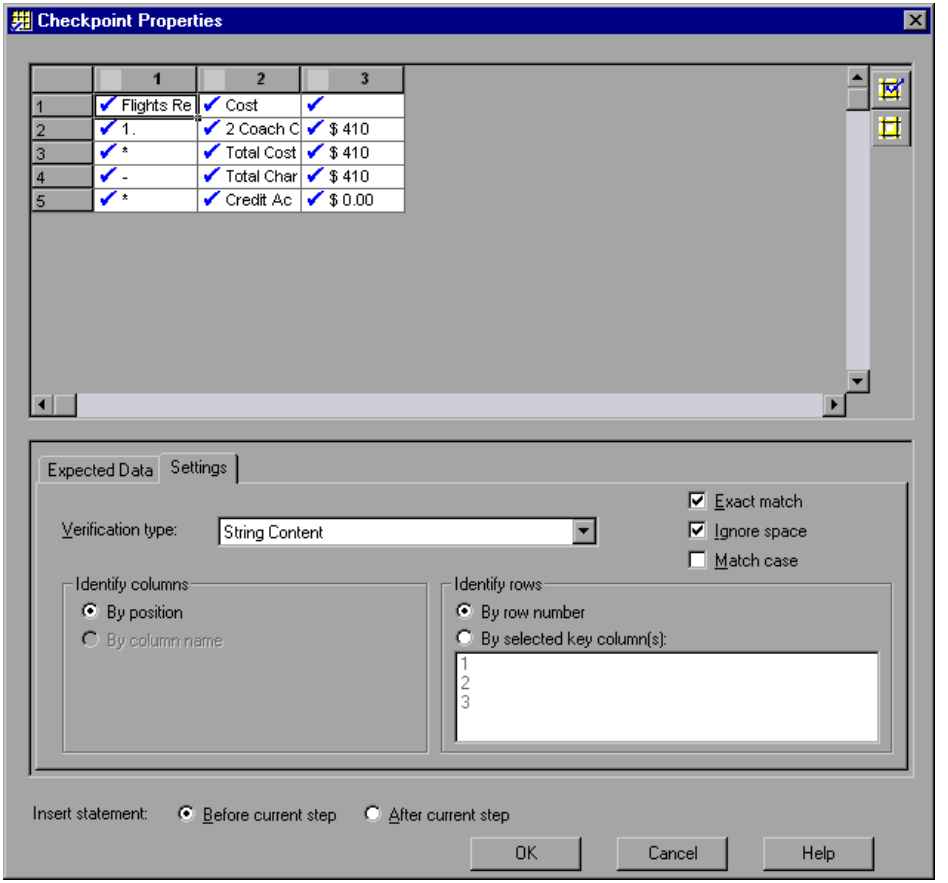
The Expected Data tab includes the following options:

Option	Description
Selected Cell	Indicates the row and column numbers of the highlighted cell in read-only format.
Constant (default)	Sets the expected value of a cell as a constant. Click the **Edit Constant Value Options** button  to open the Constant Value Options dialog box where you can set the value as a text string or a regular expression. The value of the cell is constant for each iteration of the test run.
Parameter	Sets the expected value of a cell as a parameter. For more information, see Chapter 10, "Parameterizing Tests." When you parameterize a cell, a symbol is displayed in place of its contents to denote that it is parameterized, for example:  .
Data Table	Specifies the parameter as a Data Table parameter. The value of the cell is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.
Parameter Name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.

Option	Description
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box where you can set the value as a regular expression. You can also specify the parameter name and set the parameter as global or current action data. If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box where you can specify the parameter type and preferences.

Specifying Verification Method and Type

The Settings tab displays options that determine how the actual cell contents are compared with the expected cell contents and how the cells in the table on your Web page are located. Using the following features, you can determine how QuickTest locates the cells to be checked and how to check their contents:



The Settings tab includes the following options:

Option	Description
Verification type	Specifies how cell contents are compared. • **String Content** evaluates the content of the cell as a string. For example, "2" and "2.00" are not recognized as the same string. • **Numeric Content** evaluates the content of the cell according to numeric values. For example, "2" and "2.00" are recognized as the same number. • **Numeric Range** compares the content of the cell against a numeric range. Both the minimum and maximum values are any real number that you specify. (The **Min** and **Max** fields are displayed when you select Numeric Range.) This comparison differs from text and numeric content verification in that the actual table data is compared against the range that you defined and not against the expected results.
Exact Match	Checks that the exact text, and no other text, is displayed in the cell. Clear this box if you want to check that a text string is displayed in a cell as part of the contents of the cell. **Note:** QuickTest displays this option when String Content is selected as the Verification type.
Ignore Space	Ignores spaces in the captured content when performing the check. The presence or absence of spaces does not affect the outcome of the check. **Note:** QuickTest displays this option when String Content is selected as the Verification type.
Match Case	Conducts a case sensitive check. **Note:** QuickTest displays this option when String Content is selected as the Verification type.

Option	Description
Identify columns	Specifies by column the location of the cell(s) in your table that you want to compare. • **By position** locates these cells according to the column position. A shift in the position of the columns within the table results in a mismatch. **Tip:** Select this option if your table contains multiple columns with the same name. • **By column name** locates these cells according to the column name. A shift in the position of the columns within the table does not result in a mismatch. **Note: By column name** is enabled when QuickTest recognizes a column name.
Identify rows	Specifies by row the location of the cell(s) in your table that you want to compare. • **By row number** locates these cells according to the row position. A shift in the position of any of the rows within the table results in a mismatch. • **By selected key column(s)** locates these cells by identifying the row(s) first and then the desired cell(s). QuickTest identifies rows containing a blue check mark by matching the contents of the column you select to be *key column*. Select the key column from the list box that contains the names of all columns in the table. When QuickTest identifies the row(s), it locates and compares the cell(s) corresponding to those with the blue check mark. A shift in the position of the row(s) does not result in a mismatch. If more than one row is identified, QuickTest checks the first matching row. You can use more than one key column to uniquely identify any row. **Note:** A key symbol  is displayed in the header of selected columns.

Tip: If you want to use different verification types for different parts of the table, create a separate checkpoint for each type, and specify the relevant cells for each check.

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to check the table cell before the highlighted step is performed. Choose **After current step** if you want to check the table cell after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding a table checkpoint to an existing test. It is disabled when recording a new test or modifying an existing table checkpoint.

Modifying a Table Checkpoint

You can change the expected data and settings of an existing table checkpoint.

To modify the table checkpoint:

1. In the Tree View, or the Expert View, right-click the table checkpoint that you want to modify.
2. Select **Checkpoint Properties**. The Checkpoint Properties dialog box opens.
3. To modify the settings, see the section entitled “Understanding the Checkpoint Properties Dialog Box”, on page 115.

8

Checking Databases

By adding database checkpoints to your test scripts, you can check the contents of databases accessed by your Web site.

This chapter describes:

- Creating a Check on a Database
- Understanding the Checkpoint Properties Dialog Box

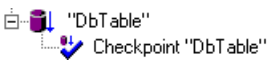
About Checking Databases

You can use database checkpoints in your test script to check databases accessed by your Web site and detect defects. You define a query on your database, and then you create a database checkpoint that checks the results of the query.

There are two ways to define a database query:

- Use Microsoft Query. You can install Microsoft Query from the *custom installation* of Microsoft Office.
- Manually define an SQL statement.

In QuickTest, you create a database checkpoint based on the results of the query you defined on a database. QuickTest captures the current information about the database and saves this information as *expected data*. A *database checkpoint* is inserted into the test script. This checkpoint is displayed in your test script as a **DbTable.Check CheckPoint** statement and as a step in the Tree View as follows:



When you run the test, the database checkpoint compares the current state of the database to the expected data defined in the Checkpoint Properties dialog box. If the expected data and the current results do not match, the database checkpoint fails. The results of the checkpoint can be viewed in the Test Results window. For more information, see Chapter 18, "Analyzing Test Results."

You can modify the expected data of a database checkpoint before or after you run your test. You can also make changes to the query in an existing database checkpoint. This can be useful if you move the database to a new location on the network.

Creating a Check on a Database

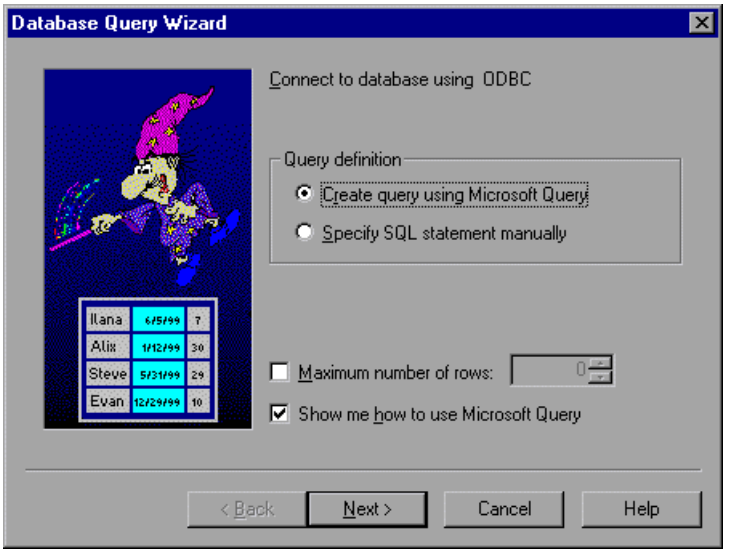
The results of a query on a database are known as a *result set*. You can create a check on a database in order to check the contents of the entire result set, or a part of it.

Creating a Database Checkpoint

You can define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement.

To create a database checkpoint:

1 Choose **Insert** > **Checkpoint** > **Database Checkpoint**. The Database Query Wizard opens.

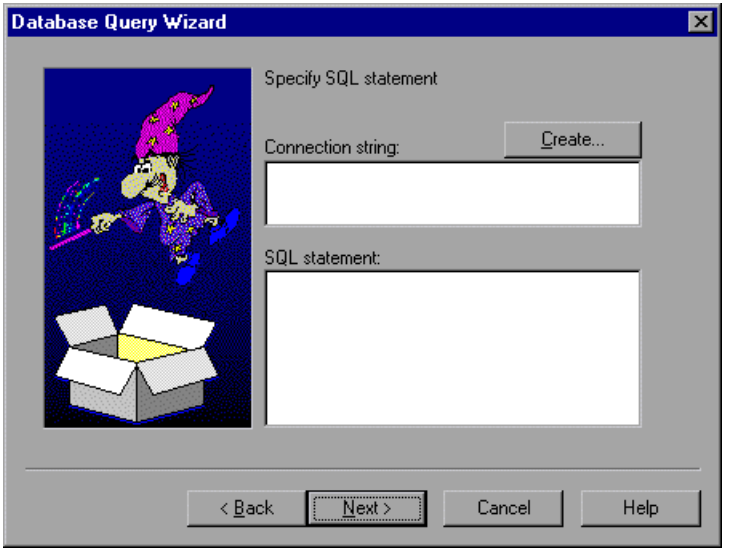


2 Select your database selection preferences and click **Next**. You can choose from the following options:

- **Create query using Microsoft Query:** Opens Microsoft Query, enabling you to create a new query. Once you finish defining your query, you return to QuickTest.
- **Specify SQL statement manually:** Opens the **Specify SQL statement** screen in the wizard, which enables you to specify the connection string and an SQL statement. For additional information, see step 3.
- **Maximum number of rows:** Select this check box if you would like to limit the number of rows, enter the maximum number of database rows to check. You can specify a maximum of 32,000 rows.
- **Show me how to use Microsoft Query:** Displays an instruction screen before opening Microsoft Query, when you click **Next**. (Enabled only when **Create query using Microsoft Query** is selected).

3 If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information about creating a query, see "Creating a Query in Microsoft Query," on page 128.

If you chose **Specify SQL statement** in the previous step, the following screen opens:



Specify the connection string and the SQL statement, and click **Next**.

- **Connection string:** Enter the connection string, or click **Create** to open the ODBC Select Data Source dialog box. You can select a **.dsn* file in the ODBC Select Data Source dialog box to have it insert the connection string in the box for you.
- **SQL statement:** Enter the SQL statement.

QuickTest takes several seconds to capture the database query and restore the QuickTest window.

4 The Checkpoint Properties dialog box opens. Select the checks to perform on the result set as described in "Understanding the Checkpoint Properties Dialog Box," on page 129. You can also modify the expected data in the result set.

5 Click **OK** to close the Checkpoint Properties dialog box. A database checkpoint is inserted in the test script.

Creating a Query in Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source. QuickTest supports the following versions of Microsoft Query:

- version 2.00 (in Microsoft Office 95)
- version 8.00 (in Microsoft Office 97)
- version 2000 (in Microsoft Office 2000)

To choose a data source and define a query in Microsoft Query:

1 When Microsoft Query opens during the Insert database checkpoint process, choose a new or an existing data source.

2 Define a query.

3 When you are done:

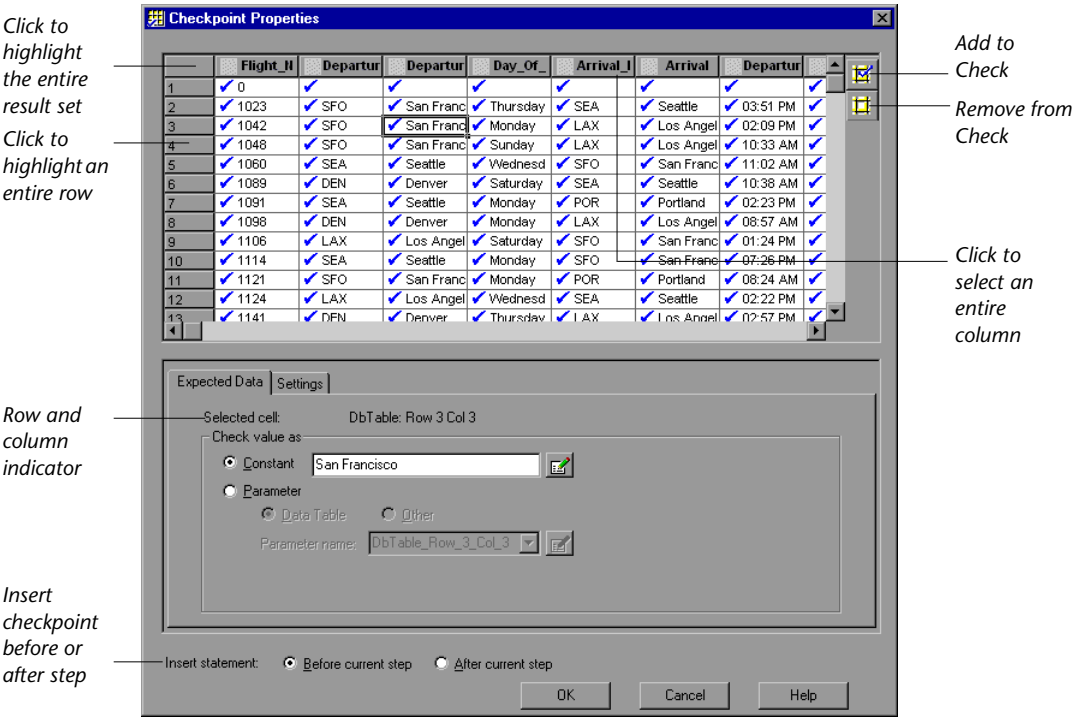
- Version 2.00: Choose **File** > **Exit and return to QuickTest** to close Microsoft Query and return to QuickTest.
- Version 8.00 and Version 2000: In the Finish screen of the Query Wizard, select **Exit and return to QuickTest** and click **Finish** to exit Microsoft Query. Alternatively, click **View data or edit query in Microsoft Query** and click **Finish**. After viewing or editing the data, choose **File** > **Exit and return to QuickTest** to close Microsoft Query and return to QuickTest.

4 Return to step 4 in the previous procedure to continue creating your database checkpoint in QuickTest.

For additional information on working with Microsoft Query, refer to the Microsoft Query documentation.

Understanding the Checkpoint Properties Dialog Box

The Checkpoint Properties dialog box enables you to specify which cells to check, and which verification method and type to use. You can also edit or parameterize the expected data for the database cells included in the check.



Specifying which Cells to Check

The top part of the Checkpoint Properties dialog box displays a grid representing the cells in the captured result set. The grid displays rows and columns of a database. You can check the entire result set, a row, a column, or a cell. QuickTest only checks cells containing a blue check mark. By default, all cells contain a check mark.

To check/remove from check:	Double-click
a single cell **Note:** In the **Check Value as** section of the Expected Data tab, the cell value is displayed in the **Constant** box.	the cell
an entire row	the row header
an entire column	the column header
the entire result set	the top-left corner of the grid

Alternatively, you can click the **Add Selection** button to check an entire selection. To remove an entire selection from the check, you can click the **Remove Selection** button.

Note: When more than one cell is selected, the options on the Expected Data tab are disabled.

Double-clicking on the grid toggles the settings of each cell in the selection.

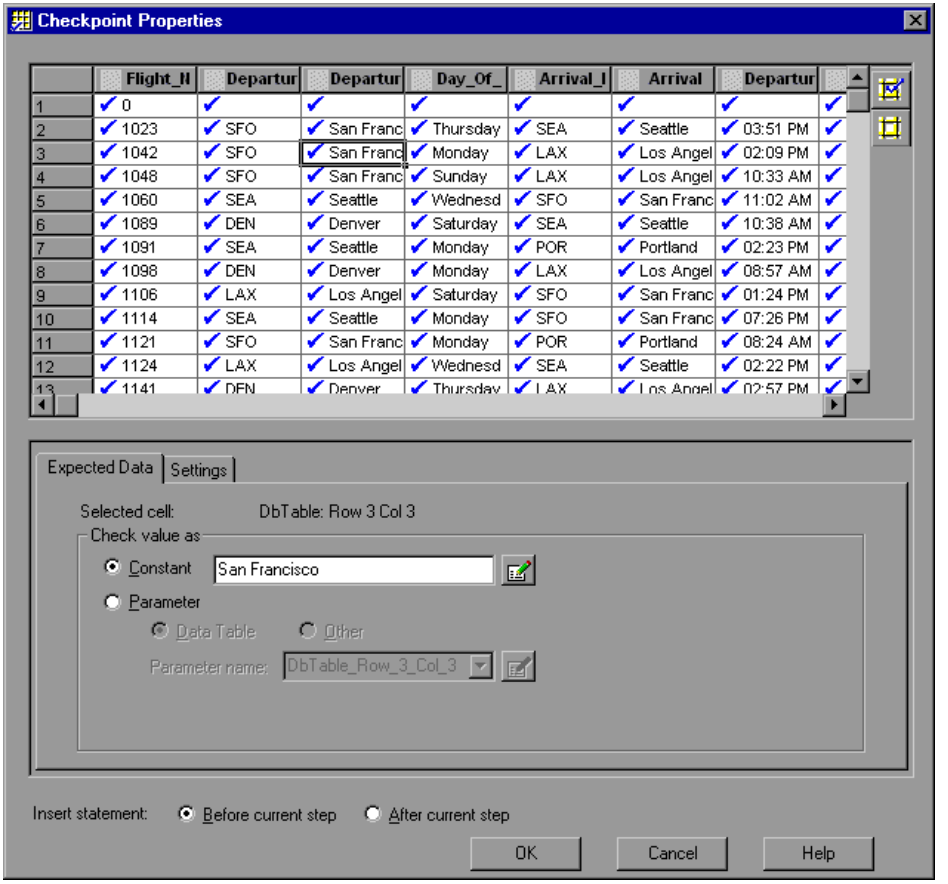
Tip: You can change the column widths and row heights of the grid by dragging the boundaries of the column and row headers.

The bottom of the Checkpoint Properties dialog box displays the **Insert Statement** option. With the **Insert Statement** option, you can set the checkpoint to occur before or after the highlighted step in your test.

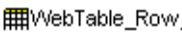
Note: The Insert Statement option is enabled when adding a database checkpoint to an existing test. It is disabled when recording a new test or modifying an existing database checkpoint.

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the highlighted cell in the result set. QuickTest compares this value with the actual value that it finds during the test run. If the expected value and the actual value do not match, the checkpoint fails. You can determine the value of the cell using the following features:



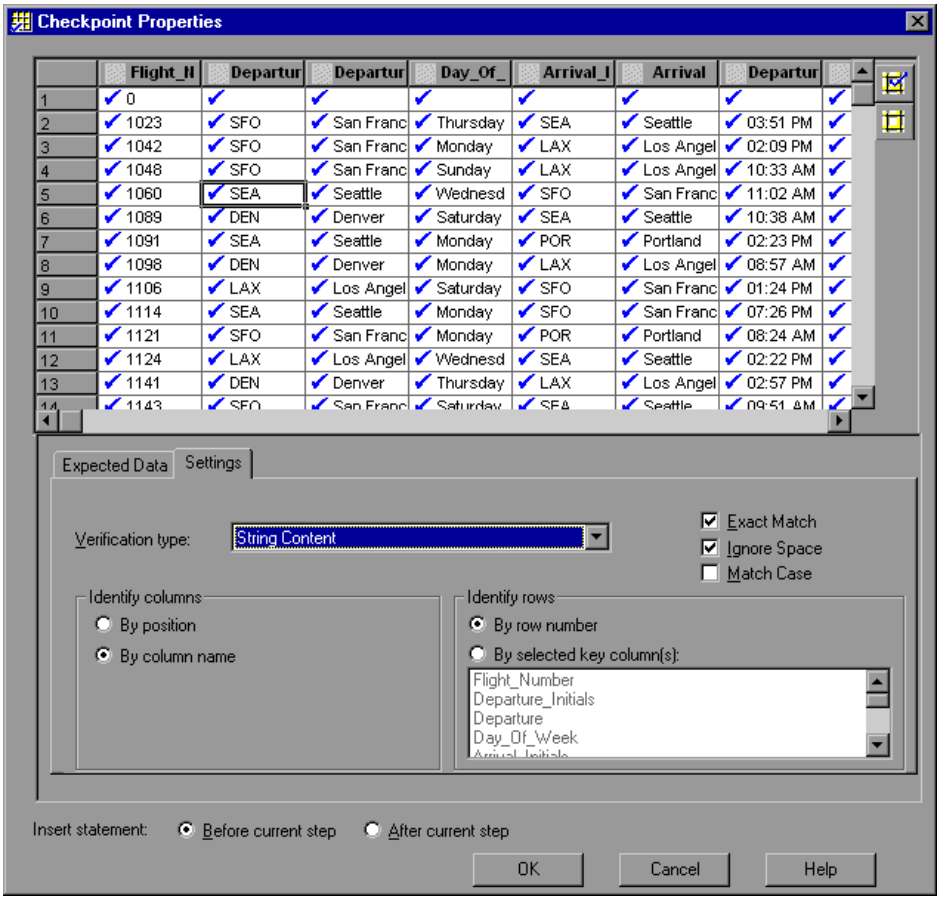
The Expected Data tab includes the following options:

Option	Description
Selected Cell	Indicates the row and column numbers of the highlighted cell in read-only format.
Constant (default)	Sets the expected value of the cell as a constant. Click the **Edit Constant Value Options** button  to open the Constant Value Options dialog box where you can set the value as a text string or a regular expression. The value of the cell is constant for each iteration of the test run.
Parameter	Sets the expected value of a cell as a parameter. For more information, see Chapter 10, "Parameterizing Tests."When you parameterize a cell, a symbol is displayed in place of its contents to denote that it is parameterized, for example:  .
Data Table	Specifies the parameter as a Data Table parameter. The value of the cell is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.

Option	Description
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box where you can set the value as a regular expression. You can also specify the parameter name and set the parameter as global or current action data. If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box where you can specify the parameter type and preferences.

Specifying the Verification Method and Type

The Settings tab displays options that determine how the actual cell contents are compared with the expected cell contents and how the cells in your result set are located. Using the following features, you can determine how QuickTest locates the cells to be checked and how to check their contents:



The default setting is a check for the exact text while ignoring spaces on the entire result set by column name and row number.

The Settings tab includes the following options:

Option	Description
Verification type	Specifies how cell contents are compared. • **String Content** evaluates the content of the cell as a string. For example, "2" and "2.00" are not recognized as the same string. • **Numeric Content** evaluates the content of the cell according to numeric values. For example, "2" and "2.00" are recognized as the same number. • **Numeric Range** compares the content of the cell against a numeric range. Both the minimum and maximum values are any real number that you specify. (The **Min** and **Max** fields are displayed when you select Numeric Range.) This comparison differs from text and numeric content verification in that the actual result set data is compared against the range that you defined and not against the expected results.
Exact Match	• Checks that the exact text, and no other text, is displayed in the cell. Clear this box if you want to check that a text string is displayed in a cell as part of the contents of the cell. **Note:** QuickTest displays this option when String Content is selected as the Verification type.
Ignore Space	• Ignores spaces in the captured content when performing the check. The presence or absence of spaces does not affect the outcome of the check. **Note:** QuickTest displays this option when String Content is selected as the Verification type.
Match Case	• Conducts a case sensitive search. **Note:** QuickTest displays this option when String Content is selected as the Verification type.

Option	Description
Identify unique columns	Specifies by column the location of the cell(s) in your database that you want to compare. • **By position** locates these cells according to the column position. A shift in the position of the columns within the database results in a mismatch. • **By column name retrieved from database** locates these cells according to the column name. A shift in the position of the columns within the database does not result in a mismatch.
Identify unique rows	Specifies by row the location of the cell(s) in your database that you want to compare. • **By row number** locates these cells according to the row position. A shift in the position of any of the rows within the database results in a mismatch. • **By selected key column(s)** locates the row(s) containing the cells to be checked by matching the value of the cell whose column was previously selected as a *key column*. A shift in the position of the row(s) does not result in a mismatch. If more than one row is identified, QuickTest checks the first matching row. You can use more than one key column to uniquely identify any row. **Note:** A key symbol  is displayed in the header of selected columns.

Tip: If you want to use different verification types for different parts of the database, create a separate checkpoint for each type, and specify the relevant cells for each check.

Modifying a Database Checkpoint

You can make the following changes to an existing database checkpoint:

- Modify the SQL query definition
- Modify the expected data, verification type or method

To modify the SQL query definition:

1. Right-click the database object that you want to modify.
2. Select **Object Properties**.
3. Modify the SQL and connection string properties as necessary and click **OK**.

To modify the expected data in a database checkpoint:

1. Right-click the database checkpoint that you want to modify.
2. Select **Checkpoint Properties**. The Checkpoint Properties dialog box opens.

   Modify the settings as described above.

9

Checking Bitmaps

QuickTest enables you to compare objects in a Web page by matching captured bitmaps.

This chapter describes:

- Checking a Bitmap
- Modifying a Bitmap Checkpoint

About Checking Bitmaps

You can check an area of your Web page as a bitmap. While creating a test, you specify the area you want to check by selecting an object. You can check an entire object or any area within an object. QuickTest captures the specified object as a bitmap, stores it in the test folder, and inserts a checkpoint in the test. When you run the test, QuickTest compares the object or selected area of the object currently displayed on the Web page with the bitmap (or selected area) stored when the test was recorded. If there are differences, QuickTest captures a bitmap of the actual object and displays it with the expected bitmap in the details portion of the Test Results window. By comparing the two bitmaps (expected and actual), you can identify the nature of the discrepancy. For more information on test results of a checkpoint, see “Viewing the Results of a Checkpoint,” on page 309.

For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. You can record the new map that is displayed after one click on the control key that zooms in the map. Using the bitmap checkpoint, you can check that the map zooms in correctly. The bitmap checkpoint can also check complex Java applets and applications whose functionality can be tested by how they are displayed on a Web page.

Checking a Bitmap

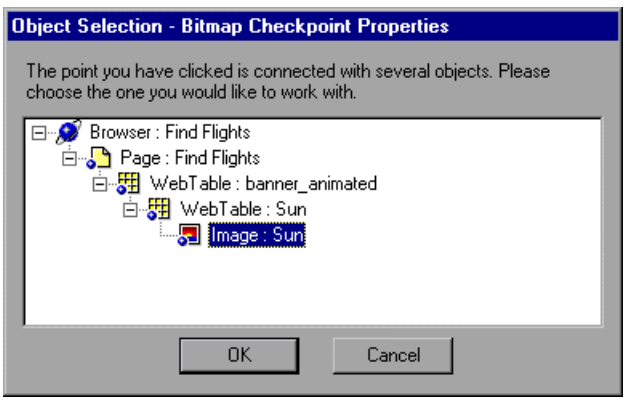
You can add a bitmap checkpoint while recording or afterward.

To create a bitmap checkpoint while recording:

1 Choose **Insert** > **Checkpoint** > **Bitmap Checkpoint** or click the arrow beside the **Insert Checkpoint** button and choose **Bitmap Checkpoint**.

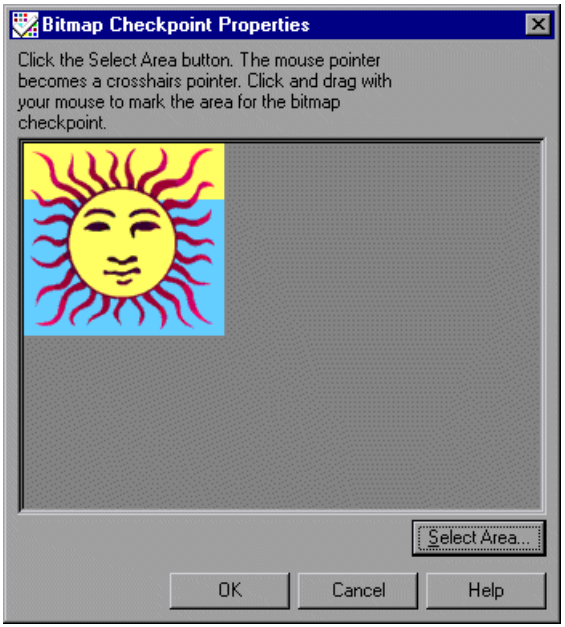
The QuickTest window is minimized, and the mouse pointer turns into a pointing hand.

2 Click an object to check in your Web page. The Object Selection - Bitmap Checkpoint Properties dialog box opens.



3 Select an object from the tree on which to create a bitmap checkpoint.

4 Click **OK**. The Bitmap Checkpoint Properties dialog box opens.



A bitmap of the object you selected in the previous step is displayed in the dialog box.

Tip: If you want to create a bitmap checkpoint of multiple objects, you should select the highest level object that includes all the objects to include in the bitmap checkpoint.

---

5 If you want to check a small area of the bitmap, click the **Select Area** button. Use the crosshairs pointer to specify a small area.

QuickTest checks the selected area while ignoring the remainder of the bitmap. The Test Results window displays the bitmap with this area highlighted.

6 Click **OK** to add the bitmap checkpoint to your test.

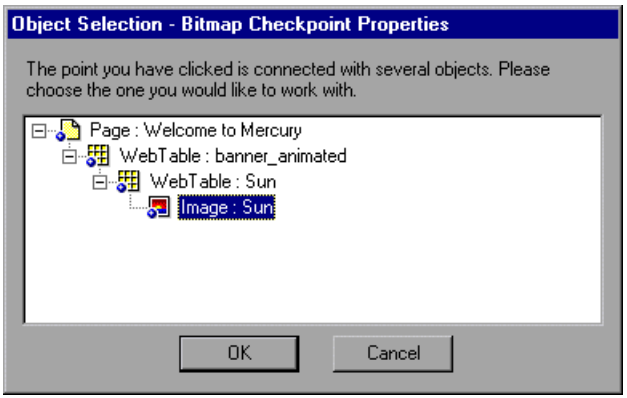
A tree item with a checkpoint icon is added to your test tree.

To create a bitmap checkpoint after recording:

1 Make sure the **Display Views** button is selected.

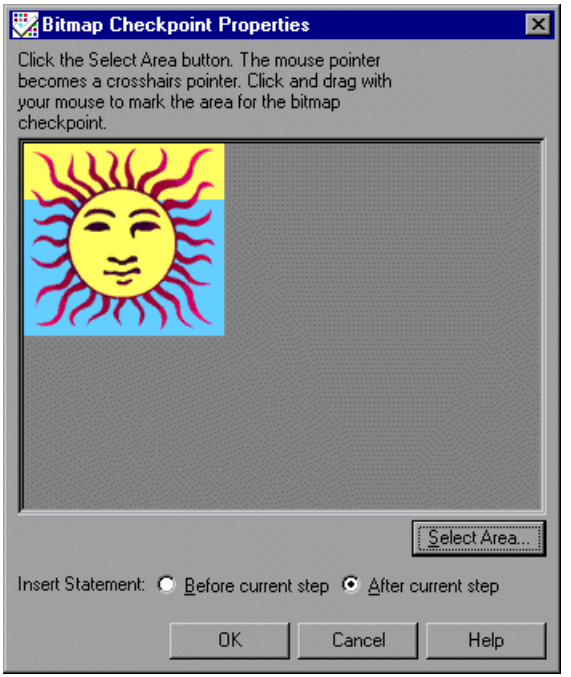
2 Click a step in the Tree View to add a checkpoint. The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click an object in the ActiveScreen and choose **Insert Bitmap Checkpoint**. The Object Selection - Bitmap Checkpoint Properties dialog box opens.



4 Select an object from the tree on which to create a bitmap checkpoint.

5 Click **OK**. The Bitmap Checkpoint Properties dialog box opens.



A bitmap of the object you selected in the previous step is displayed in the dialog box.

Tip: If you want to create a bitmap checkpoint of multiple objects, you should select the highest level object that includes all the objects to include in the bitmap checkpoint.

---

6 If you want to check a small area of the bitmap, click the **Select Area** button. Use the crosshairs pointer to specify a small area.

QuickTest checks the selected area while ignoring the remainder of the bitmap. The Test Results window displays the bitmap with this area highlighted.

7 Choose to insert the bitmap checkpoint before or after the highlighted step.

Note: Choose **Before current step** if you want to check the bitmap before the highlighted step is performed. Choose **After current step** if you want to check the bitmap after the highlighted step is performed.

The Insert Statement option is enabled when adding a bitmap checkpoint to an existing test. It is disabled when recording a new test or modifying an existing bitmap checkpoint.

---

8 Click **OK** to add the bitmap checkpoint to your test.

A tree item with a checkpoint  icon is added to your test tree.

Modifying a Bitmap Checkpoint

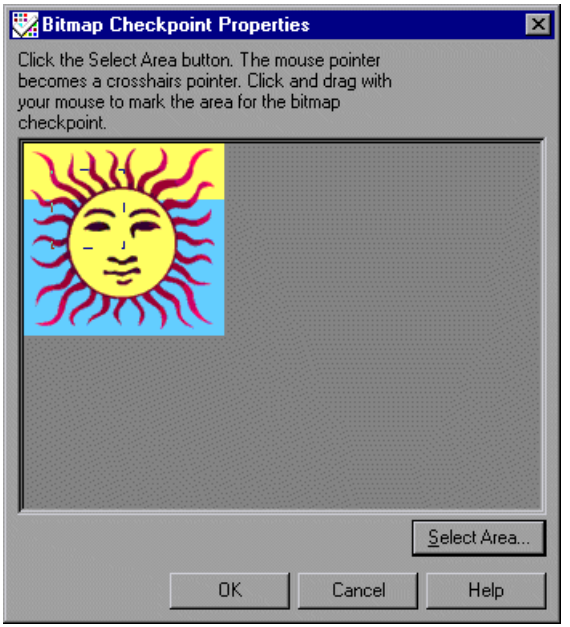
You can modify an existing bitmap checkpoint:

To modify a bitmap checkpoint:

1 In the Test pane, right-click the bitmap checkpoint that you want to modify and select **Checkpoint Properties**.

Tip: You can also double-click the checkpoint in the Test pane to open the Bitmap Checkpoint Properties dialog box.

The Bitmap Checkpoint Properties dialog box opens.



2 Click **Select Area**. Use the crosshairs pointer to specify the area of the bitmap you want to check.

   QuickTest checks the selected area while ignoring the remainder of the bitmap. The Test Results window displays the bitmap with this area highlighted.

3 Click **OK** to modify the checkpoint.

10

Parameterizing Tests

QuickTest enables you to expand the scope of a basic test by replacing fixed values with parameters. This process, known as *parameterization*, greatly increases the power and flexibility of your tests.

This chapter describes:

- Parameterizing Your Test
- Parameter Types
- Example of a Parameterized Test

About Parameterizing Tests

You can use QuickTest to enhance your tests by parameterizing values in the test. A *parameter* is a variable that is assigned a value from various data sources or generators.

There are two different groups of parameters: data table parameters and all other parameters.

About Data Table Parameters

Data table parameters enable you to create a *data-driven test* (or action) that runs several times using the data you supply. In each repetition, or *iteration*, QuickTest substitutes the constant value with a different value from the data table.

About Other Parameters

Other parameter types enable you to use variable values from other sources during the test run. These values may be values you supply or values that QuickTest generates for you based on conditions and options you choose.

For example, you can have QuickTest insert a random number in an edit field, or you can have QuickTest read all the values for filling in a Web form from an external user-defined file. You can also use one of QuickTest's built-in environment variables to insert current information about the test or the machine running the test.

Parameterizing Your Test

When you add parameters to your test, you can parameterize a step recorded in your test or a checkpoint added to your test.

You parameterize steps, and checkpoints by opening the appropriate dialog box as described in this section. Once you have opened the appropriate dialog box, follow the instructions in "Parameterizing Steps and Checkpoints," on page 150.

You can parameterize a step in your test tree while recording your test or afterward. A step is made up of an *object* that you navigate in your Web page, and/or a *method* by which you navigate the step. When you parameterize a step, you must choose whether you want to parameterize an object property or a method argument.

For example, suppose your Web site includes an application that enables users to search for contact information from a membership database. The application displays the user's contact information and a button that says "View <MemName>'s Picture", where <MemName> is the name of the member that the user entered. You can parameterize the button's name property so that during each iteration of the test run, QuickTest can identify the picture button.

Your Web site may also include a form with an edit field into which the user types a text string. You may want to test how your site responds to different data in the form. Rather than record a separate test for each text string

typed, you can parameterize the **Set** method so that during each iteration of the test run, QuickTest sets a different text string into the edit field.

You parameterize object properties in the Object Properties dialog box or the Object Repository dialog box.

To open the Object Properties dialog box, right-click a step in the test tree and choose **Object Properties**. The Object Properties dialog box opens and displays the properties of the object in the step you want to parameterize. For more information about the Object Properties dialog box, see Chapter 5, "Creating Tests."

To open the Object Repository dialog box, right-click an action containing an object with the property or value you want to parameterize, and choose **Object Repository**, or choose **Tools** > **Object Repository**.

Note: When you parameterize an object property, all occurrences of the specified object within the action are parameterized. For more information about actions, see Chapter 14, "Working with Actions."

You parameterize method arguments in the Method Arguments dialog box. To open the Method Arguments dialog box, right-click a step in the test tree and choose **Method Arguments**. The Method Arguments dialog box opens and displays the method arguments in the step.

When you parameterize a checkpoint, instead of checking how your Web site performs an operation on a single text string or object, you can check how it performs the same operation with multiple sets of data.

For example, if you are testing the sample flight Web site, "Mercury Tours," you may create a checkpoint to check that once you book a ticket, it is booked correctly. Suppose that you want to check that flights are booked correctly for a variety of different destinations. Rather than create a separate test with a separate checkpoint for each destination, you can parameterize the destination information: for each iteration of the test, QuickTest checks the flight information for a different destination.

You parameterize checkpoints in the Checkpoint Properties dialog box. To parameterize an existing checkpoint, right-click the checkpoint in the test tree and choose **Checkpoint Properties**.

- For a page checkpoint, the Page Checkpoint Properties dialog box opens.
- For a text checkpoint, the Text Checkpoint Properties dialog box opens.
- For an object checkpoint, the Checkpoint Properties dialog box opens.
- For a table checkpoint, the Checkpoint Properties dialog box opens.
- For a database checkpoint, the Checkpoint Properties dialog box opens.

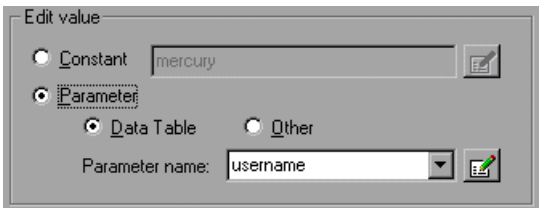
For more information on checkpoints, and for information on parameterizing checkpoints while creating them, see Chapter 6, "Creating Checkpoints."

Parameterizing Steps and Checkpoints

You can parameterize a step or checkpoint while recording your test or afterward.

To parameterize a step or checkpoint:

1. Open the Object Properties, Object Repository, Method Arguments, or Checkpoint Properties dialog box for the step or checkpoint you want to parameterize as described in the previous section.
2. Select the value to parameterize (if applicable).
3. In the Edit value box, click **Parameter**.



4. Click **Data Table** or **Other** to indicate the type of parameter you want to use.

For more information about parameter types, see "Parameter Types," on page 151.

5 If you chose **Data Table**, choose a parameter from the **Parameter Name** box list or enter a new name.

- To use an existing parameter, select it from the list.
- To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter.

6 If you chose **Other**, or if you want to modify the default data table parameter options, click the **Edit Parameter Options** button to set your desired preferences.

7 Click **OK** to save your changes and close the Parameter Options dialog box.

For more information on the parameter options for each parameter type, see "Parameter Types," on page 151.

8 Click **OK** to save the parameter and close the dialog box. A new column is highlighted in the table for the new parameter.

In your test tree, the  icon next to the step indicates that the step has been parameterized.

Note: You can specify additional data values for Data Table parameters by entering them directly into the table in the Data pane. For more information, see Chapter 15, "Working with Data Tables."

Parameter Types

You can parameterize your test using Data Table parameters, or by having QuickTest insert values for you based on the parameter type and the parameter-specific preferences you set.

The following parameter types are available:

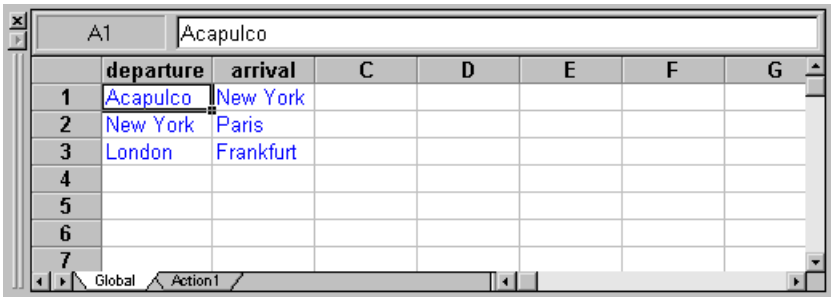
- Data Table Parameters
- Environment Variable Parameters
- Random Number Parameters

Note: In addition to the procedures described below, you can also define data table, environment, and random number variables using parameterization objects and methods in the Expert View. For more information, refer to the *QuickTest Object Model Reference*.

Data Table Parameters

You can supply the list of possible values for a parameter by creating a data table parameter. Data table parameters enable you to create a data-driven test (or action) that runs several times using the data you supply. In each repetition, or *iteration*, QuickTest substitutes the constant value with a different value from the data table.

For example, consider the sample Web site, "Mercury Tours," which enables you to book flight requests. To book a flight, you supply the flight itinerary and click the **Continue** button. The site returns the available flights for the requested itinerary. You could conduct the test by accessing the Web site and recording the submission of numerous queries. This is a slow, laborious, and inefficient solution. When you parameterize your test, you first record a test that accesses the Web site and checks for the available flights for one requested itinerary. You then substitute the recorded itinerary with a data table parameter and add your own multiple sets of data, one for each itinerary, in the Data pane.

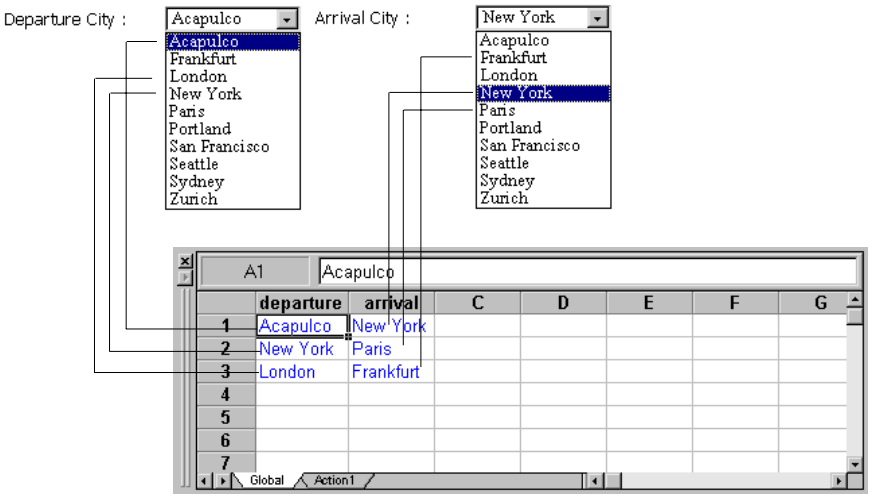


Each *column* in the table represents the list of values for a single parameter. The column header is the parameter name.

Each *row* in the table represents a set of values that QuickTest submits for all the parameters during a single iteration of the test. When you run your test, QuickTest runs one iteration of the test for each row of data in the table. Thus, a test with ten rows in the data table will run ten times.

Tip: You can also create output values, which retrieve variables from the test while it runs and insert them into a column in the data table. You can then use these columns as data table parameters in your test. For more information, see Chapter 11, "Creating Output Values."

In the previous example, QuickTest submits a separate query for each itinerary when you run the test.

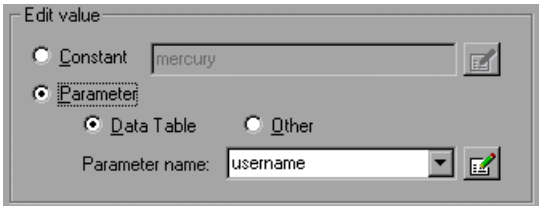


When you create a data table parameter, a new column is added in the Data Table and the current value of the property or argument you parameterized is placed in the first row.

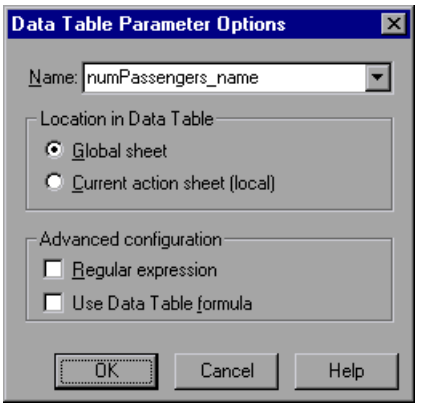
By default, the data table parameter is placed in the global data table and the values are taken as simple text. You can modify these settings in the Parameter Options dialog box.

To modify Data Table Parameter options:

1 In the **Edit value** section of the Object Properties, Object Repository, Method Arguments, or Checkpoint Properties dialog box, click **Data Table** as the type of parameter you want to use.



2 Click the **Edit Parameter Options** button next to the **Parameter name** box. The Data Table Parameter Options dialog box opens.



3 Choose a parameter from the **Name** box list or enter a new name.

- To use an existing parameter, select it from the list.
- To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter.

4 Click **Global sheet** or **Current action sheet (local)**.

- To add the parameter to the Global tab in the Data pane, click **Global sheet**.
- To add the parameter to the Action tab, click **Current action sheet (local)**.

For more information about the Global vs. Current Action Data Table, see "Setting Data Table Parameters as Global or Action" below. For more information about actions, see Chapter 14, "Working with Actions."

5 If you want to set the property value of a checkpoint as a Data Table formula, select the **Use Data Table Formula** box. For more information, see Chapter 15, "Working with Data Tables."

6 If you want to set the property value of a checkpoint as a regular expression, select the **Regular expression** check box (where applicable). For more information, see Chapter 12, "Using Regular Expressions."

Setting Data Table Parameters as Global or Action

When you parameterize a step using the DataTable, you must decide whether you want to make it a *global parameter* or an *action parameter*.

Global parameters take data from the global sheet in the data table. The global sheet contains the data that replaces global parameters in each iteration of the test. By default, the test runs one iteration for each row in the global sheet of data table. You can also set the test to run only one iteration or to run iterations on specified rows within the global sheet of the data table.

For more information about setting global iteration preferences, see "Run Test Settings," on page 409.

Action parameters take data from the action's sheet in the data table. The data in the action's sheet replaces the action's parameters in each iteration of the action. By default, actions run only one iteration. You can also set the test to run iterations for all rows in the action's sheet, or to run iterations on specified rows within the action's sheet.

For more information about setting action iteration preferences, see "Setting the Run Properties for an Action," on page 254.

Note: After running a parameterized test, you can view the results of values taken from the DataTable in the Runtime Data table. For more information, see "Viewing the Runtime Data Table" on page 311.

Environment Variable Parameters

QuickTest can insert a value from the an Environment variable list, which is a list of variables and corresponding values that can be accessed from your test. Throughout the test run, the value of an environment variable remains the same, regardless of the number of iterations, unless you change the value of the variable programmatically in your script.

Tip: The environment parameter is especially useful for localization testing, when you want to test an application where the user interface strings change depending on the selected language.

You can also vary the input values for each language by selecting a different data table file each time you run the test. For more information, see Chapter 15, "Working with Data Tables."

There are three types of environment variables:

- **User-Defined**
  - **Internal** - variables that you define within the test. They are accessible only within the test in which they were defined.
  - **External** - variables that you have defined in the active external environment variables file. You can create as many files as you want and select an appropriate file for each test. Note that external environment variable values are read-only within the test.
- **Built-in** - read-only, built-in variables such as Test path and Operating system. They are accessible from all tests.

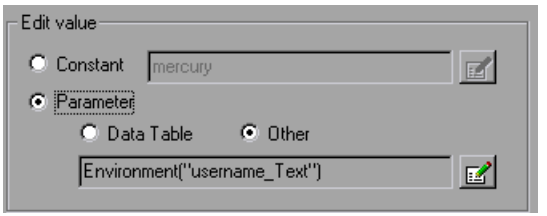
Using Internal User-Defined Environment Variables

You can add or modify internal user-defined environment variables for your test in the Parameter Options dialog box or in the Environment tab of the Test Settings dialog box.

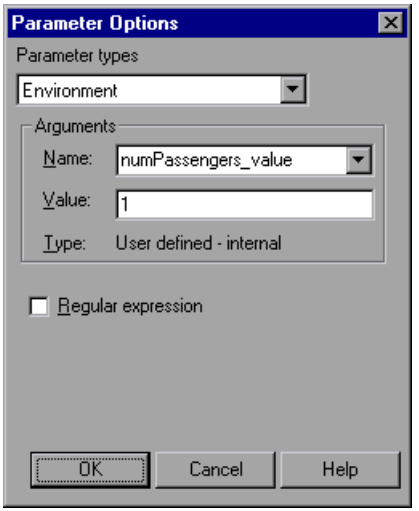
The section below describes how to add or modify environment variables from the Parameter Options dialog box. For more information on adding or modifying environment variables in the Test Settings dialog box, see “Environment Test Settings,” on page 414.

To add or modify environment variable parameters in the Parameter Options dialog box:

1 In the **Edit value** section of the Object Properties, Object Repository, Method Arguments, or Checkpoint Properties dialog box, click **Other** as the type of parameter you want to use.



2 Click the **Edit Parameter Options** button next to the parameter type box. The Parameter Options dialog box opens.



3 Select **Environment** as the **Parameter Type**.

4 Accept the default name or enter a new name to add a new internal environment parameter, or select an existing environment variable name from the **Name** box. If you select an existing internal parameter, you can modify the value.

5 If you created a new parameter, or selected an existing internal parameter, enter the value for the parameter in the **Value** box.

6 If you want to set the property value of a checkpoint as a regular expression, select the **Regular Expression** check box. For more information, see Chapter 12, "Using Regular Expressions."

Note: If you select an external user-defined or a built-in environment variable name, the Value box and the Regular Expression check box are disabled. The variable type of the selected parameter is displayed below the **Value** box.

---

7 Click **OK** to save your changes and close the dialog box.

Using External User-Defined Environment Variable Files

You can create a list of variable-value pairs in an external file written in *ini* file format, and use variables from the file as parameters.

Tip: You can create several external variable files with the same variable names and different values, and run the test several times - using a different file each time. This is especially useful for localization testing.

To create an external environment variables file:

1 Open any text editor.

2 Type [Environment] on the first line.

3 Type one variable-value pair on each line in the format: variable=value.

4 Save the file in a location that is accessible from the QuickTest machine. The file must be a text file, but you can use any file name extension.

For example:

```
[Environment]
MyParam1=10
MyParam2=20
MyParam3=Happy Birthday
```

Note: You can also export all existing user-defined variables (internal and external) to a new external environment variables file. For more information, see Chapter 25, "Setting Testing Options for a Single Test."

To select the active external environment variables file:

1 Choose **Test** > **Settings** to open the Test Settings dialog box.

2 Click the **Environment** tab.

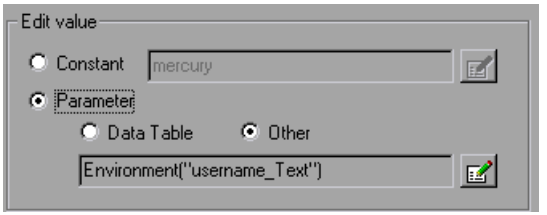
3 Select the **Retrieve variables and values from file (reloaded each test run)** check box.

4 Use the browse button, or enter the full path of the external environment variables file you want to use with your test.

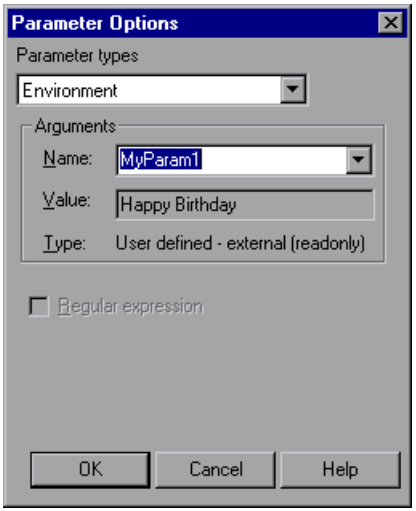
For more information about the Test Settings dialog box, see Chapter 25, "Setting Testing Options for a Single Test."

To insert an external user-defined environment variable:

1 In the **Edit value** section of the Object Properties, Object Repository, Method Arguments, or Checkpoint Properties dialog box, click **Other** as the type of parameter you want to use.



2 Click the **Edit Parameter Options** button next to the parameter type box. The Parameter Options dialog box opens.



3 Select **Environment** as the **Parameter Type**.

4 Select the external user-defined parameter you want to use from the **Name** box. The variable type of the selected parameter is displayed below the **Value** box. The current value is displayed in the **Value** box.

Note: If you modify the external user-defined parameter name, you create a new internal user-defined parameter. The external parameter still exists and cannot be modified from this dialog box.

---

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

Using Built-in Environment Variables

QuickTest provides a set of built-in variables that enable you to use current information about the test and the QuickTest machine running your test, such as the test name, the test path, the operating system type and version, and the local host name.

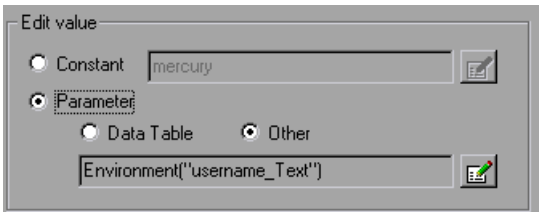
For example, you may want to perform different checks in your test based on the operating system being used by the computer that is running the test. To do this, you could include the **OSVersion** built-in environment variable in an **If** statement.

The following built-in environment variables are available:

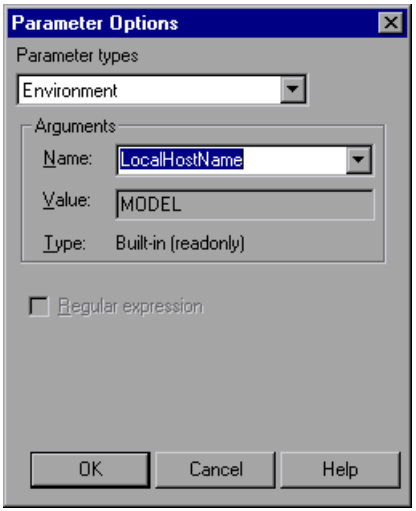
Name	Description
LocalHostName	Local host name
OS	Operating system
OSVersion	Operating system version
ResultDir	Path of the folder in which the current test results are located
SystemTempDir	System temporary directory
TestDir	Path of the folder in which the test is located
TestName	The name of the test
UserName	Windows login user name

To insert a built-in environment variable:

1 In the **Edit value** section of the Object Properties, Object Repository, Method Arguments, or Checkpoint Properties dialog box, click **Other** as the type of parameter you want to use.



2 Click the **Edit Parameter Options** button next to the parameter type box. The Parameter Options dialog box opens.



3 Select **Environment** as the **Parameter Type**.

4 Select the built-in parameter you want to use from the **Name** box. The variable type of the selected parameter is displayed below the **Value** box. The current value is displayed in the **Value** box.

Note: If you modify the built-in parameter name, you create a new internal parameter. The built-in parameter still exists and cannot be modified.

---

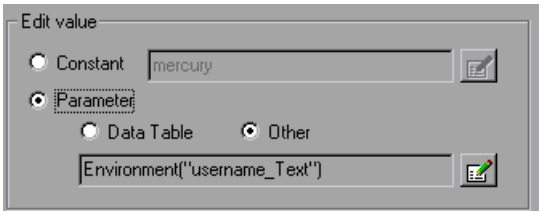
5 Click **OK** to close the dialog box.

Random Number Parameters

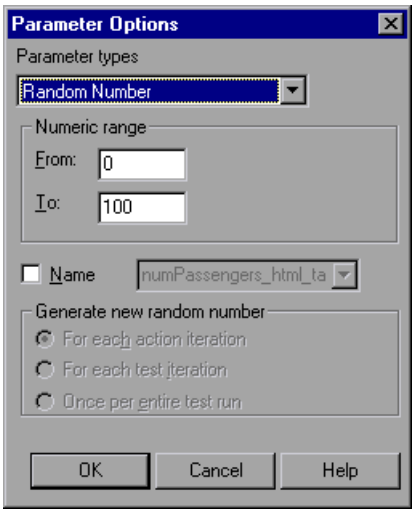
QuickTest can generate random numbers and insert them as the values for a parameter. By default, the random number ranges between 0-100, and a different random number is generated each time the parameter is called. You can modify these settings in the Parameter Options dialog box.

To modify Random Number Parameter options:

1 In the **Edit value** section of the Object Properties, Object Repository, Method Arguments, or Checkpoint Properties dialog box, click **Other** as the type of parameter you want to use.



2 Click the **Edit Parameter Options** button next to the parameter type box. The Parameter Options dialog box opens.



3 Select **Random Number** as the type of parameter you want to use.

4 Enter the start and end of the range for the random number selection in the **From** and **To** boxes, respectively.

5 If you want to insert the same parameter more than once within your test, select the **Name** box.

Assigning a name to a random number parameter enables you to use the same parameter several times in your test. If the parameter is used in the test

more than before the Generate New Random Number event has occurred (as described in step 7), the same number is used. If the parameter is used again after the event has occurred, a new number is generated based on the same numeric range.

6 If you selected the **Name** box, choose an existing parameter from the **Parameter Name** box list or enter a new name.

- To use an existing parameter, select it from the list.
- To create a new parameter, enter a descriptive name for the parameter.

Note: If you select an existing parameter, then changing the settings in the dialog box affects all instances of that parameter in the test.

---

7 If you selected the **Name** box, choose the timing for generation of a new random number:

- **For each action iteration**- A new number is generated for each action iteration.
- **For each test iteration** - A new number is generated for each global iteration.
- **Once per entire test run**- A new number is generated at the beginning of each test run. The same number is used throughout the test run.

Note: The number of action and global iterations performed during the test run is based on the number of rows in the data table.

---

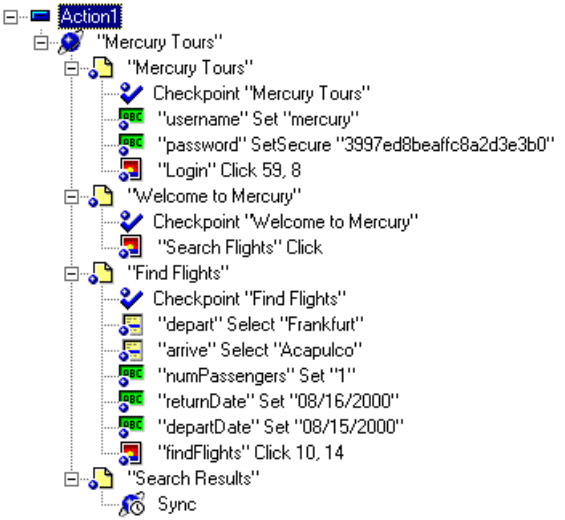
8 Click **OK** to save your changes and close the dialog box.

Example of a Parameterized Test

The following example shows how to parameterize a step object, step method, and checkpoint using data table parameters.

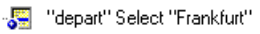
When you test your Web site, you may want to check how it performs the same operations with multiple sets of data. For example, if you are testing the sample flight Web site, "Mercury Tours," you may want to check that the correct departure and the arrival cities are selected before you book a particular flight. Suppose that you want to check that the flights are booked correctly for a variety of different locations. Rather than create a separate test with a separate checkpoint for each location, you can parameterize the location information: for each iteration of the test, QuickTest checks the flight information for a different location.

The following is a sample test of a flight booking procedure. The departure city is "Frankfurt" and the arrival city is "Acapulco."

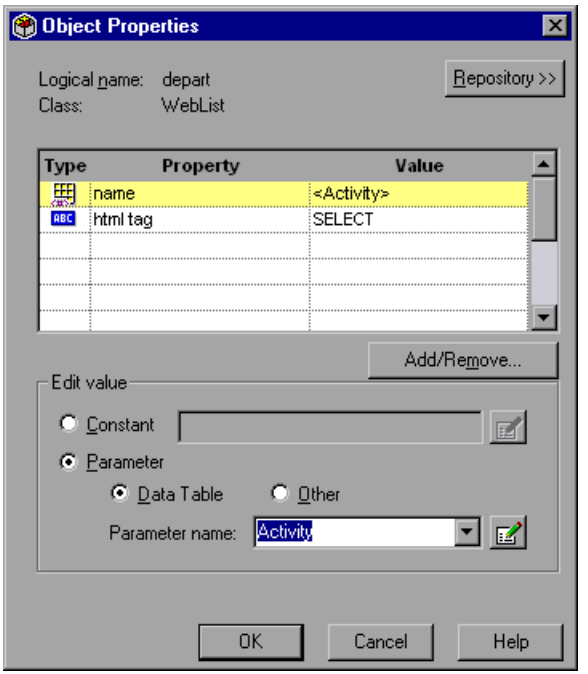


Parameterize a Step

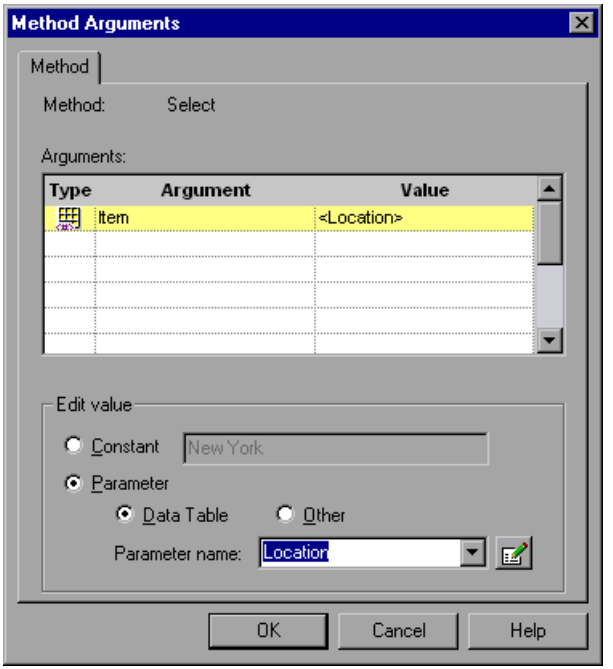
Parameterize the object and the method of the following step:



In the Object Properties dialog box, select the “name” property. Select the **Parameter** radio button. In the Parameter name box, rename depart_name to Activity. Close the dialog box. The Activity column is added to the data table.



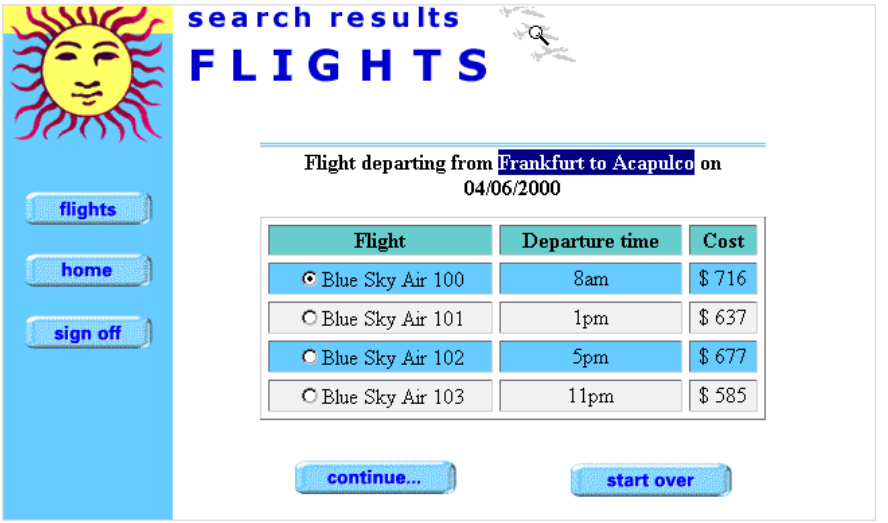
In the following example, parameterize the method of the above step. In the Method Arguments dialog box, select the "item" property. Select the **Parameter** radio button. In the Parameter name box, rename depart_item to Location. Close the dialog box. The Location column is added to the data table.



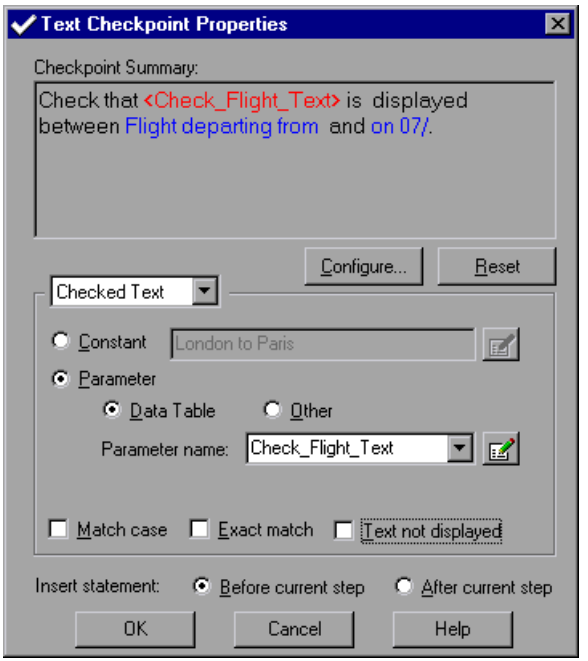
For more information on parameterizing a step, see "Parameterizing Your Test" on page 148.

Parameterize a Checkpoint

In the following example, a parameterized text checkpoint is added to check that the correct locations were selected before you book a flight. A text checkpoint is created for the highlighted text:



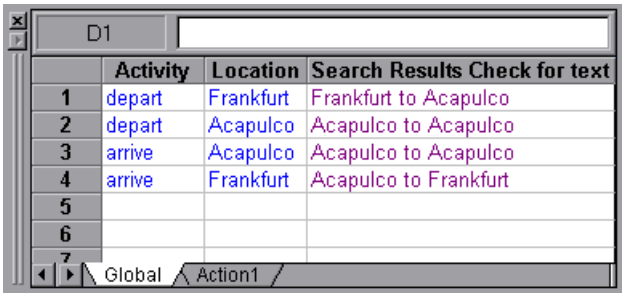
In the Text Checkpoint Properties dialog box, select **Parameter** to parameterize the selected text. Click **OK**. A text checkpoint parameter column is added to the data table.



For more information on parameterizing a checkpoint, see "Parameterizing Your Test" on page 148.

Enter Data in the Data Table

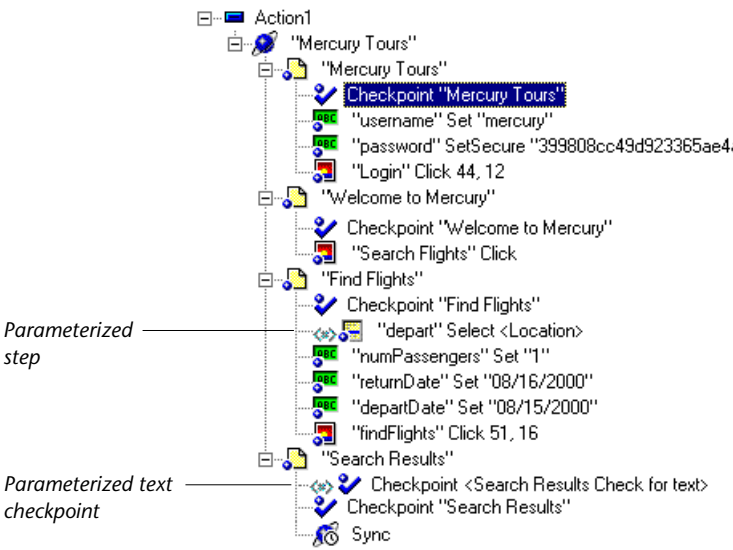
Complete the table in the Data pane. The data table may be displayed as follows:



For more information on data tables, see Chapter 15, “Working with Data Tables.”

Modified Test

The following example shows the test after parameterizing the step and creating a parameterized text checkpoint.



11

Creating Output Values

QuickTest enables you to retrieve a value from your test and store it in the table in the Data pane as an output value. You can subsequently use this output value as an input variable in your test. This enables you to use data retrieved during a test in other parts of the test.

This chapter describes:

- Creating Page Output Values
- Creating Text Output Values
- Creating Object Output Values
- Creating Image Output Values
- Creating Table Output Values
- Creating Database Output Values

About Creating Output Values

You can add output values to your test. An *output value* is a value captured during the test run and stored in the data table for use at another point in the test run. When you create an output value, the test retrieves a value at the specified point during the test run and stores it in a column in the data table. When the value is needed later in the test run as input, you can retrieve it from the data table. You can create output values from property values for Web pages, objects, and images. You can also create output values from text strings and the contents of tables and databases.

You can use checkpoints, method arguments, or the Object Properties dialog box to retrieve the output value from the data table and input it during the test run where it is required.

For example, consider a flight reservation site. You design a test to create a new reservation and then view the reservation details. Every time you run the test, the site generates a unique order number for the new reservation. To view the reservation, the site requires the user to input the same order number. You cannot know the order number before you run the test.

To solve this problem, you create an output value for the unique order number that the site generates when creating a new reservation. In the View Reservation screen, you insert the output value into the order number input field. You use the data table column containing the unique order number as a parameter.

When you run the test, QuickTest retrieves the unique order number generated by the site for the new reservation and stores it in the data table as an output value. When the test reaches the order number input field required to view the reservation, QuickTest inputs the unique order number stored in the data table into the order number field.

You can insert an output value by using commands on the Insert menu or by clicking the arrow beside the Insert Checkpoint button on the Main toolbar. This displays a menu of options that are relevant to the selected step in the test tree.

Note: After running your test, you can view the output values retrieved during a test run in the Runtime Data table. For more information, see "Viewing the Runtime Data Table" on page 311.

Creating Page Output Values

You can create a page output value from a page property value. When you run the test, QuickTest retrieves the current value of the property and stores it in the data table as an output value. For example, the number of links on a Web page may vary based on the selections a user makes on a form on the previous page. You could make an output value to store the number of links on the page during each test run or iteration. To create a page output value, you open the Page Output Value Properties dialog box.

Adding a Page Output Value

You can create a page output value while recording your test or afterward.

To create a page output value while recording:

1 Choose **Insert** > **Output Value** > **Standard Output Value**.

The mouse pointer turns into a pointing hand.

2 Click the Web page. The Object Selection - Output Value Properties dialog box opens.

3 Select the **Page** item and click **OK**. The Page Output Value Properties dialog box opens.

4 Specify the settings for the output value. For more information, see “Understanding the Page Output Value Properties Dialog Box” on page 177.

5 Click **OK** to close the Page Output Value Properties dialog box.

An output value tree item  is added to your test tree.

To create a page output value after recording:

1 Make sure the **Display Views** button is selected.

2 Click a page step in your test tree where you want to add an output value. The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click the page in the ActiveScreen and choose **Insert Output Value**. The Object Selection - Output Value Properties dialog box opens.

4 Select the **Page** item and click **OK**. The Page Output Value Properties dialog box opens.

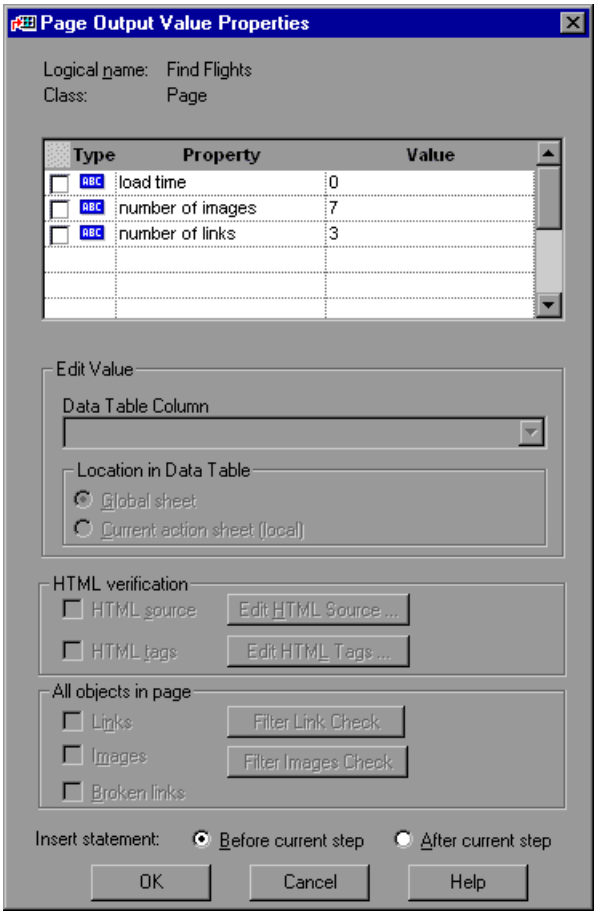
5 Specify the settings for the output value. For more information, see "Understanding the Page Output Value Properties Dialog Box" on page 177.

6 Click **OK** to close the Page Output Value Properties dialog box.

An output value tree item  is added to your test tree.

Understanding the Page Output Value Properties Dialog Box

In the Page Output Value Properties dialog box, you can choose which property of the page to specify as an output value.



Identifying the Page

The top part of the dialog box displays information about the page:

Information	Description
Logical name	The name of the page as defined in the HTML code of the Web page.
Class	The type of object. This is always Page.

Choosing which Property to Specify as an Output Value

The dialog box also displays the page properties that you can specify as an output value, in a pane that lists the properties, their values, and their types:

Pane Element	Description
Check box	To specify a property as an output value, select the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently an output value.
Property	The name of the property.
Value	The current value of the property. If you specify the property as an output value, the value column displays a data table column name.

Specifying the Settings for the Output Value

In the Edit Value section, you use the following options to specify the output value settings:

Option	Description
Data Table Column	Specifies the data table column name. Use the default name, or type a descriptive name.
Global sheet (default)	Adds the data table column name to the Global tab in the Data pane. For more information, see Chapter 14, "Working with Actions."
Current action sheet (local)	Adds the data table column name to the current action sheet in the Action tab in the Data pane. For more information, see Chapter 14, "Working with Actions."

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to output the value of the page property before the highlighted step is performed. Choose **After current step** if you want to output the value of the page property after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding a page output value to an existing test. It is disabled when recording a new test or modifying an existing page output value.

Creating Text Output Values

You can create a text output value from a text string. When you run the test, QuickTest retrieves the current value of the text string and stores it in the data table as an output value. To create a text output value, you open the Text Output Value Properties dialog box.

Adding a Text Output Value

You can create a text output value while recording your test or afterward.

To create a text output value while recording:

1 Highlight the text string you want to specify as an output value.

2 Choose **Insert** > **Output Value** > **Text Output Value**.

The mouse pointer turns into a pointing hand.

3 Click the text string you want to specify as an output value. The Text Output Value Properties dialog box opens.

4 Specify the settings for the output value. For more information, see "Understanding the Text Output Value Properties Dialog Box" on page 181.

5 Click **OK** to close the Text Output Value Properties dialog box.

An output value tree item is added to your test tree.

To create a text output value after recording:

1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to create an output value.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Highlight and then right-click the text string to specify as an output value in the ActiveScreen.

4 Choose **Insert Text Output**. The Text Output Value Properties dialog box opens.

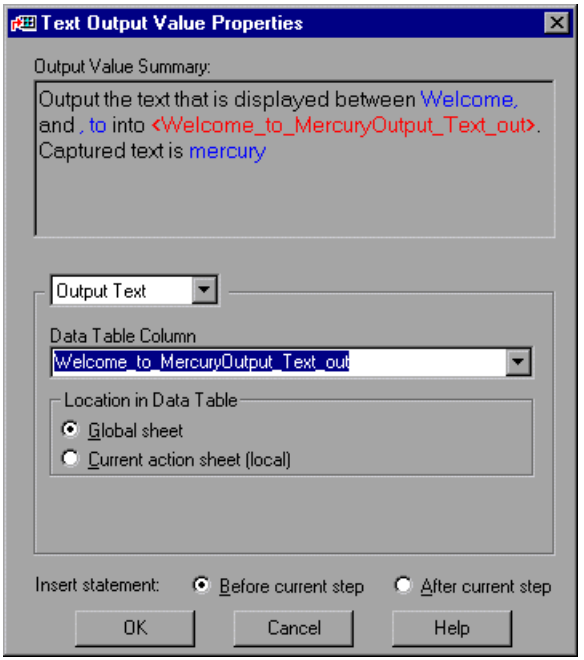
5 Specify the settings for the output value. For more information, see "Understanding the Text Output Value Properties Dialog Box" on page 181.

6 Click **OK** to close the Text Output Value Properties dialog box.

An output value tree item is added to your test tree.

Understanding the Text Output Value Properties Dialog Box

In the Text Output Value Properties dialog box, you can specify a text string as an output value as well as which text is displayed before and after the output value text string. This is particularly helpful when the text string you want to specify as an output value is displayed several times in the same Web page.



The top of the dialog box displays a pane that summarizes where the output value text string is located.

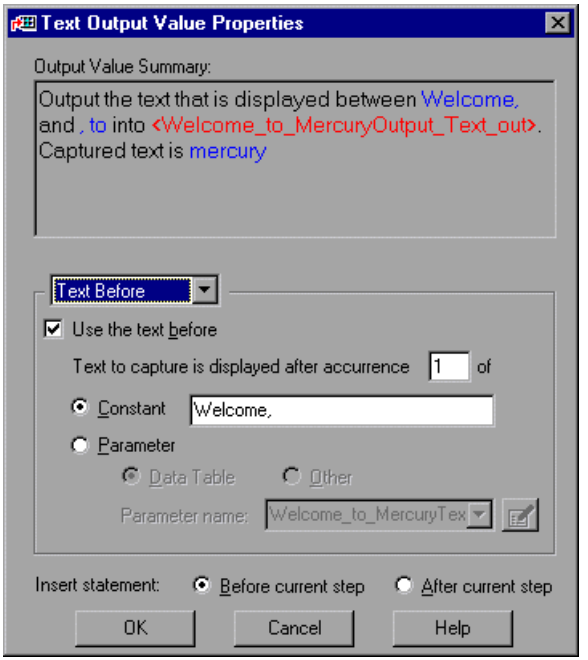
Specifying Text as an Output Value

Choose **Output text** from the list box. The following options are displayed for specifying the output value settings:

Option	Description
Data Table Column	Specifies the data table column name. Use the default name, or type a descriptive name.
Global sheet (default)	Adds the data table column name to the Global tab in the Data pane. For more information, see Chapter 14, "Working with Actions."
Current action sheet (local)	Adds the data table column name to the current action sheet in the Action tab in the Data pane. For more information, see Chapter 14, "Working with Actions."

Specifying the Text Displayed Before the Text Output Value

Choose **Text Before** from the list box.



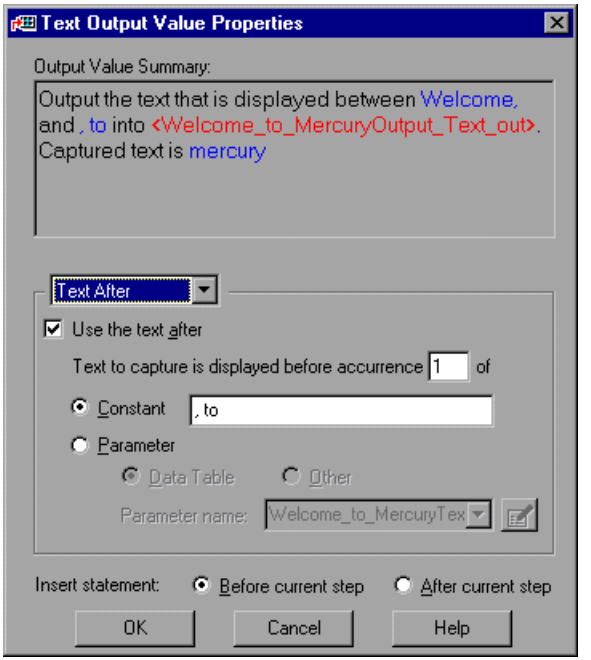
The following options are displayed for specifying which text, if any, is displayed before the output value string:

Option	Description
Use the text before	If selected, enables the options below. If cleared, QuickTest looks for all occurrences of the output value text string while ignoring the text displayed before it.
Text to capture is displayed after occurrence __ of (default)	Outputs the text string displayed after the text specified in the Data Table Column box. If the identical text you specify is displayed more than once on the page, you can specify to which occurrence of the text you are referring. If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate. If you choose non-unique text, change the occurrence number appropriately. For example, if you want to output the text string displayed after the third occurrence of the words "Mercury Tours", enter 3 in the **Text to capture is displayed after occurrence** box.
Constant (default)	Sets the expected value of the text before the captured text as a constant. The value is constant for each iteration of the test run. Note that the text box displays the text before the captured text. You can change the text by typing in the text box. **Tip:** If you modify the text, use a unique string whenever possible so that the occurrence number is 1.
Parameter	Sets the expected value of the text before the captured text as a parameter. For more information, see Chapter 10, “Parameterizing Tests.”

Option	Description
Data Table	Specifies the parameter as a Data Table parameter. The value of the text before the captured text is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the DataTable Parameter Options dialog box, where you can also specify the parameter name and set the parameter as global or current action data. If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

Specifying the Text Displayed After the Output Value Text

Choose **Text After** from the list box.



The following options are displayed for specifying which text, if any, is displayed after the output value text string:

Option	Description
Use the text after	If selected, enables the options below. If cleared, QuickTest looks for all occurrences of the output value text string while ignoring the text displayed after it.
Text to capture is displayed before occurrence __ of (default)	Outputs the text string displayed before the specified text in the Data Table Column box. QuickTest starts counting occurrences of the **is displayed before** text you specify, from the end of the **is displayed after** string. In other words, it starts looking for the specified text from the text string you selected to output. If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the recommended text and the text you specify is displayed in the text string to output as well as in the **is displayed before** text, you need to modify the occurrence number accordingly.
Constant (default)	Sets the expected value of the text after the captured text as a constant. The value is constant for each iteration of the test run. Note that the text box displays the text after the captured text. You can change the text by typing in the text box. **Tip:** If you modify the text, use a unique string whenever possible so that the occurrence number is 1.
Parameter	Sets the expected value of the text after the captured text as a parameter. For more information, see Chapter 10, "Parameterizing Tests."
Data Table	Specifies the parameter as a Data Table parameter. The value of the text after the captured text is determined by the data in the Data pane. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.

Option	Description
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit Parameter Options 	If you select the **Data Table** option, the Edit Parameter Options button opens the Data Table Parameter Options dialog box, where you can also specify the parameter name and set the parameter as global or current action data. If you select the **Other** option, the Edit Parameter Options button opens the Parameter Options dialog box, where you can specify the parameter type and preferences.

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to output the value of the text string before the highlighted step is performed. Choose **After current step** if you want to output the value of the text string after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding a text output value to an existing test. It is disabled when recording a new test or modifying an existing text output value.

Creating Object Output Values

You can create an object output value from an object property value. When you run the test, QuickTest retrieves the current value of the property and stores it in the data table as an output value. To create an object output value, you open the Output Value Properties dialog box.

Adding an Object Output Value

You can create an object output value while recording your test or afterward.

To create an object output value while recording:

1 Choose **Insert** > **Output Value** > **Standard Output Value**.

The mouse pointer turns into a pointing hand.

2 Click the object. The Object Selection - Output Value Properties dialog box opens.

3 Select the tree item for which you want to specify an output value. The tree item name depends on the object's class, for example:

Icons	Object	Class
	Check box	WebCheckBox
	Edit box	WebEdit
	Radio button	WebRadioGroup
	List box	WebList
	Element	WebElement

4 Click **OK**. The Output Value Properties dialog box opens.

5 Specify the settings for the output value. For more information, see "Understanding the Output Value Properties Dialog Box" on page 190.

6 Click **OK** to close the Output Value Properties dialog box.

An output value tree item  is added to your test tree.

To create an object output value after recording:

1. Make sure the **Display Views** button is selected.
2. Click a step in your test where you want to create an output value.

   The ActiveScreen displays the Web page corresponding to the highlighted step.
3. Right-click the object for which you want to specify an output value in the ActiveScreen and choose **Insert Output Value**. The Object Selection - Output Value Properties dialog box opens.
4. Select the tree item you want to specify for an output value. The tree item name depends on the object's class, for example:

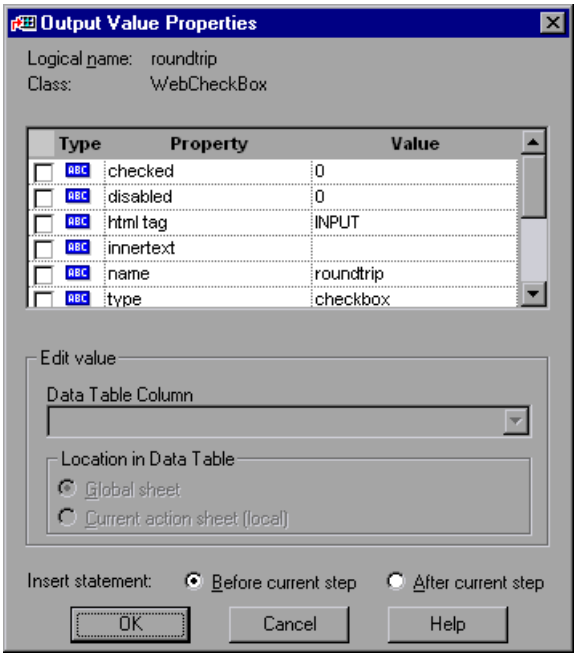
Icons	Object	Class
	Check box	WebCheckBox
	Edit box	WebEdit
	Radio button	WebRadioGroup
	List box	WebList
	Element	WebElement

5. Click **OK**. The Output Value Properties dialog box opens.
6. Specify the settings for the output value. For more information, see "Understanding the Output Value Properties Dialog Box" on page 190.
7. Click **OK** to close the Output Value Properties dialog box.

   An output value tree item  is added to your test tree.

Understanding the Output Value Properties Dialog Box

In the Output Value Properties dialog box, you can choose which property of the object to specify as an output value.



Identifying the Object

The top part of the dialog box displays information about the object:

Information	Description
Logical name	The name of the object as defined in the HTML code of the Web page.
Class	The type of object. In this example, the "WebCheckBox" class indicates that the object is a check box.

Choosing which Property to Specify as an Output Value

The dialog box also displays the object properties that you can specify as an output value, in a pane that lists the properties, their values, and their types:

Pane Element	Description
Check box	To specify a property as an output value, select the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently an output value.
Property	The name of the property.
Value	The value of the property. If you specify the property as an output value, the Value box displays the data table column name.

Specifying the Settings for the Output Value

In the Edit Value section, you use the following options to specify the output value settings:

Option	Description
Data Table Column	Specifies the data table column name. Use the default name, or type a descriptive name.
Global sheet (default)	Adds the data table column name to the Global tab in the Data pane. For more information, see Chapter 14, “Working with Actions.”
Current action sheet (local)	Adds the data table column name to the current action sheet in the Action tab in the Data pane. For more information, see Chapter 14, “Working with Actions.”

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to output the value of the property before the highlighted step is performed. Choose **After current step** if you want to output the value of the property after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding an output value to an existing test. It is disabled when recording a new test or modifying an existing output value.

Creating Image Output Values

You can create an image output value from an image property value. When you run the test, QuickTest retrieves the current value of the property and stores it in the data table as an output value. To create an image output value, you open the Image Output Value Properties dialog box.

Adding an Image Output Value

You can create an image output value while recording your test or afterward.

To create an image output value while recording:

1 Choose **Insert** > **Output Value** > **Standard Output Value**.

The mouse pointer turns into a pointing hand.

2 Click the image. The Object Selection - Output Value Properties dialog box opens.

3 Select the **Image** item you want to specify for an output value.

4 Click **OK**. The Image Output Value Properties dialog box opens.

5 Specify the settings for the output value. For more information, see "Understanding the Image Output Value Properties Dialog Box" on page 194.

6 Click **OK** to close the Image Output Value Properties dialog box.

An output value tree item is added to your test tree.

To create an image output value after recording:

1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to create an output value.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click the image for which you want to specify an output value in the ActiveScreen and choose **Insert Output Value**. The Object Selection - Output Value Properties dialog box opens.

4 Select the **Image** item you want to specify for an output value.

5 Click **OK**. The Image Output Value Properties dialog box opens.

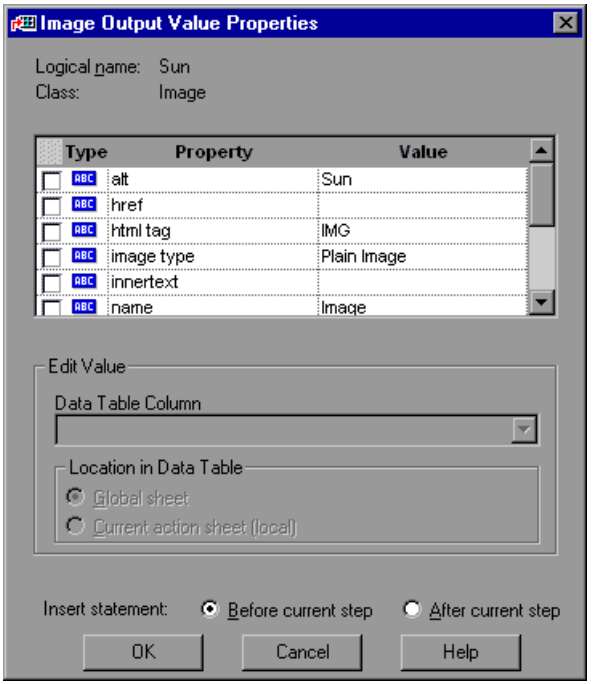
6 Specify the settings for the output value. For more information, see "Understanding the Image Output Value Properties Dialog Box" on page 194.

7 Click **OK** to close the Image Output Value Properties dialog box.

An output value tree item is added to your test tree.

Understanding the Image Output Value Properties Dialog Box

In the Image Output Value Properties dialog box, you can choose which property of the image to specify as an output value.



Identifying the Image

The top part of the dialog box displays information about the image:

Information	Description
Logical name	The name of the image as defined in the HTML code of the Web page.
Class	The type of object. This is always Image.

Choosing which Property to Specify as an Output Value

The dialog box also displays the image properties that you can specify as an output value, in a pane that lists the properties, their values, and their types:

Pane Element	Description
Check box	To specify a property as an output value, select the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently an output value.
Property	The name of the property.
Value	The value of the property. If you specify the property as an output value, the Value box displays the data table column name.

Specifying the Settings for the Output Value

In the Edit Value section, you use the following options to specify the output value settings:

Option	Description
Data Table Column	Specifies the data table column name. Use the default name, or type a descriptive name.
Global sheet (default)	Adds the data table column name to the Global tab in the Data pane. For more information, see Chapter 14, “Working with Actions.”
Current action sheet (local)	Adds the data table column name to the current action sheet in the Action tab in the Data pane. For more information, see Chapter 14, “Working with Actions.”

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to output the value of the image property before the highlighted step is performed. Choose **After current step** if you want to output the value of the image property after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding an image output value to an existing test. It is disabled when recording a new test or modifying an existing image output value.

Creating Table Output Values

You can create a table output value from the contents of a table cell. When you run the test, QuickTest retrieves the current value of a table cell and stores it in the data table as an output value. To create a table output value, you open the Output Value Properties dialog box.

Adding a Table Output Value

You can create a table output value while recording your test or afterward.

To create a table output value while recording:

1 Choose **Insert** > **Output Value** > **Standard Output Value**.

The mouse pointer turns into a pointing hand.

2 Click the table. The Object Selection - Output Value Properties dialog box opens.

3 Select a **WebTable** item and click **OK**. The Output Value Properties dialog box opens.

4 Specify the settings for the output value. For more information, see “Understanding the Output Value Properties Dialog Box” on page 198.

5 Click **OK** to close the Output Value Properties dialog box.

An output value tree item  is added to your test tree.

To create a table output value after recording:

1 Make sure the **Display Views** button is selected.

2 Click a step in your test where you want to create an output value.

The ActiveScreen displays the Web page corresponding to the highlighted step.

3 Right-click the table you want to specify as an output value in the ActiveScreen and choose **Insert Output Value**. The Object Selection - Output Value Properties dialog box opens.

4 Select a **WebTable** item and click **OK**. The Output Value Properties dialog box opens.

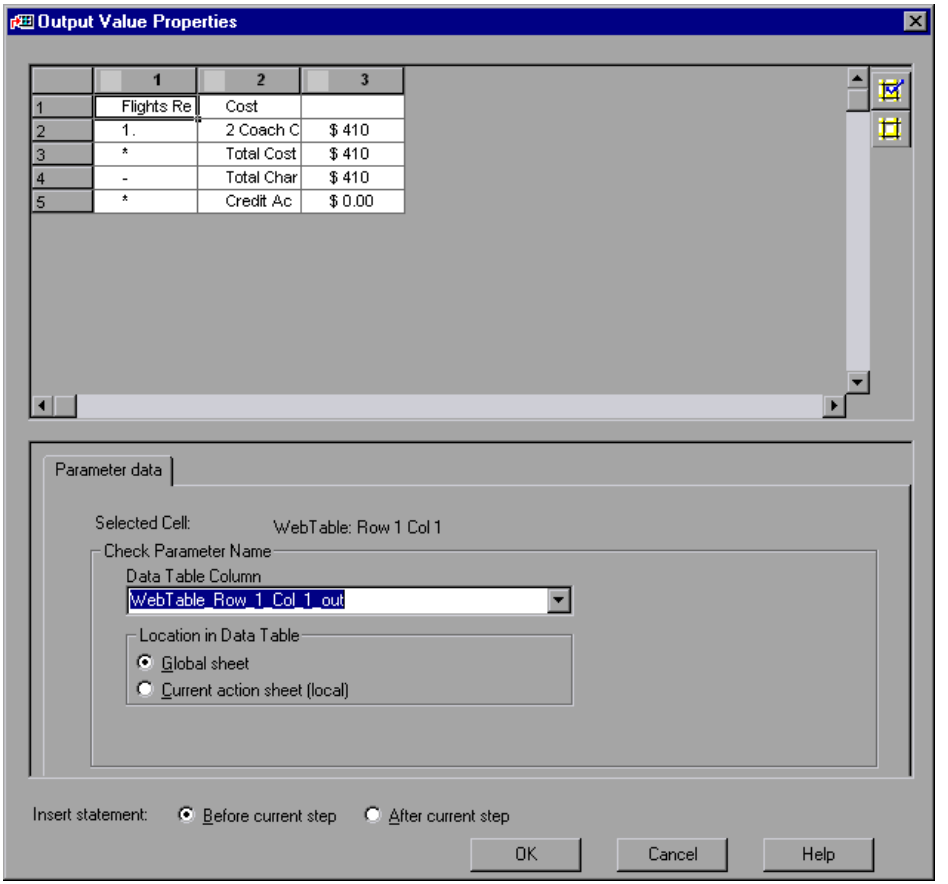
5 Specify the settings for the output value. For more information, see "Understanding the Output Value Properties Dialog Box" on page 198.

6 Click **OK** to close the Output Value Properties dialog box.

An output value tree item is added to your test tree.

Understanding the Output Value Properties Dialog Box

In the Output Value Properties dialog box, you can specify for which cells to create output values.



Choosing Cells to Specify as Output Values

The top part of the dialog box displays a grid representing the cells in a captured table. The grid displays the rows and columns of the table. You can specify one or more cells as output values. Each cell specified as an output value has a data table column. QuickTest outputs the contents of cells containing an output value icon.

To specify a cell as an output value, double-click the cell. An output value icon is added to the cell next to its captured contents.


Alternatively, you can click the **Add Output Value** button to specify a cell as an output value. To remove a cell from an output value, you can click the **Remove Output Value** button.

Tip: You can change the column widths and row heights of the grid by dragging the boundaries of the column and row headers.

Specifying the Settings for the Output Value

Use the following options to specify the output value settings:

Option	Description
Selected Cell	Indicates the row and column number of the highlighted cell in the grid at the top of the dialog box.
Data Table Column	Specifies the data table column name. Use the default name, or type a descriptive name.
Global sheet (default)	Adds the data table column name to the Global tab in the Data pane. For more information, see Chapter 14, "Working with Actions."
Current action sheet (local)	Adds the data table column name to the current action sheet in the Action tab in the Data pane. For more information, see Chapter 14, "Working with Actions."

The bottom part of the dialog box displays the Insert Statement option. Choose **Before current step** if you want to output the value of the contents of the table cell before the highlighted step is performed. Choose **After current step** if you want to output the value of the contents of the table cell after the highlighted step is performed.

Note: The Insert Statement option is enabled when adding a table output value to an existing test. It is disabled when recording a new test or modifying an existing table output value.

Creating Database Output Values

You can create a database output value from the contents of a database cell. When you run the test, QuickTest retrieves the current value of a database cell and stores it in the data table as an output value. To create a database output value, you open the Output Value Properties dialog box.

Adding a Database Output Value

You can create a database output value while recording your test or afterward.

To create a database output value while recording or afterward:

1 Choose **Insert** > **Output Value** > **Database Output Value**.

The Database Query wizard opens.

2 To create the output value using the wizard, follow the instructions for creating a database checkpoint in "Creating a Check on a Database," on page 125.

The Output Value Properties dialog box opens.

3. Specify the settings for the output value. For more information, see “Understanding the Output Value Properties Dialog Box” on page 198.
4. Click **OK** to close the Output Value Properties dialog box.

   An output value tree item  is added to your test tree.

12

Using Regular Expressions

You can use regular expressions to increase the flexibility and adaptability of your tests.

This chapter describes:

- Using Regular Expressions in Steps
- Using Regular Expressions in Object Checkpoints
- Using Regular Expressions in Text Checkpoints
- Regular Expression Syntax

About Regular Expressions

When you run your test, regular expressions enable QuickTest to identify Web objects and text strings with varying values. You can use regular expressions when defining the properties of a step, when parameterizing a step, and when creating checkpoints with varying values. For example, when you create a checkpoint on a text string with a varying date, you can define the date as a regular expression.

A regular expression is a string that specifies a complex search phrase. By using special characters such as a period (.), asterisk (*), caret (^), and brackets ([ ]), you define the conditions of the search. When one of these special characters is preceded by a backslash (\), QuickTest searches for the literal character.

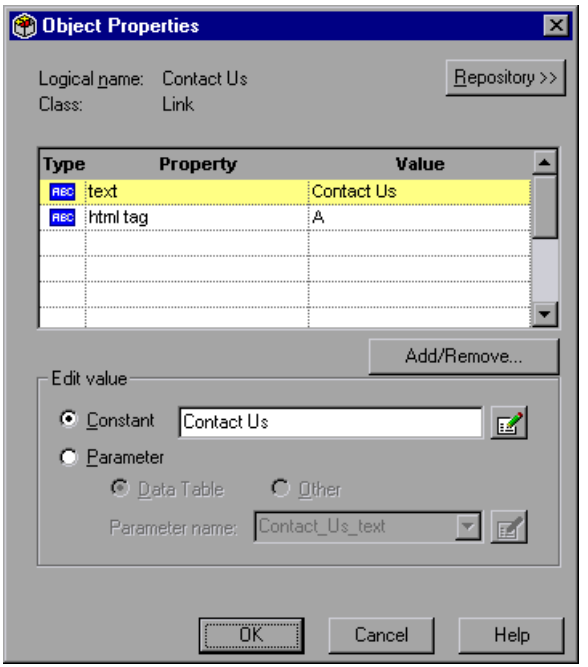
Using Regular Expressions in Steps

You can use regular expressions when defining or parameterizing a step in your test tree. A step is made up of an *object* in your Web page or application, and/or a *method* by which you navigate the step. If you expect the value of an object property value to change during each test run, you can use regular expressions when defining or parameterizing the object of a step.

For example, your site may include a form in which the user inputs data and clicks the Send button to submit the form. When a required field is not completed, the form is displayed again for the user to complete. When resubmitting the form, the user clicks the Resend button. You can define the value of the button's "name" property as a regular expression, so that QuickTest ignores variations in the button name when clicking the button.

To define a property value as a regular expression:

1 Right-click a step in the test tree and choose **Object Properties**. The Object Properties dialog box opens.



2 In the **Property** section, click the property you want to set as a regular expression. The property is highlighted.

3 In the **Edit value** section, click **Constant**.

4 Click the **Edit Constant Value Options** button. The Constant Value Options dialog box opens.

In the **Value** box, enter the regular expression syntax for the string. For information on regular expression syntax, see "Regular Expression Syntax," on page 218.

5 Select the **Regular Expression** check box.

6 You are prompted to add a backslash (\) before each special character in the **Value** text box if you want to treat it literally.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), hyphen (-), asterisk (*), caret (^), brackets ([ ]), parentheses (()), dollar sign ($), vertical line (|), plus sign (+), question mark (?), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as literal character.

- Click **Yes** to instruct QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
- Click **No** to instruct QuickTest to treat all the characters literally, except for special characters.

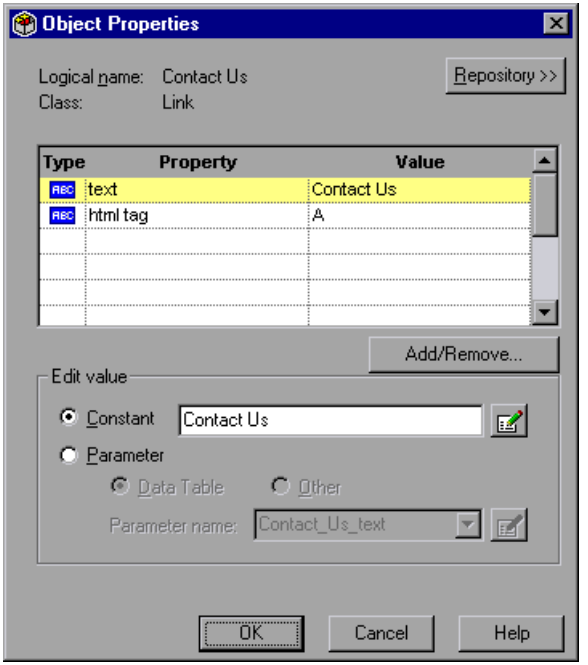
7 Click **OK** to close the Constant Value Options dialog box.

8 Click **OK** to save and close the Object Properties dialog box.

To parameterize a property value using regular expressions:

1 Right-click a step in the test tree and choose **Object Properties**.

The Object Properties dialog box opens.



2 In the **Properties** section, click the property you want to set as a regular expression. The property is highlighted.

3 In the **Edit value** section, click **Parameter**. Specify the type of parameter you want:

➤ Click **Data Table** to create a data table parameter. Click the **Edit Parameter Options** button to open the **DataTable Parameter Options** dialog box.

In the **Name** box, choose a parameter from the list or enter a new name. To use a parameter that you already created, select it from the list. To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter. For more information on parameterization, see Chapter 10, "Parameterizing Tests."

To add the parameter to the Global tab in the Data pane, select **Global sheet**. To add the parameter to the Action tab, select **Current action sheet (local)**. For more information about Actions, see Chapter 14, "Working with Actions."

Note: The value in the **Name** box should not be defined as a regular expression. When you add additional values for the parameter into the table in the Data pane, you specify the values as regular expressions.

For information on regular expression syntax, see "Regular Expression Syntax," on page 218. For information on parameterization, see Chapter 10, "Parameterizing Tests."

➤ Click **Other** to create a parameter type other than a data table parameter. Click the **Edit Parameter Options** button to open the **Parameter Options** dialog box.

Choose **Environment**. Accept the default name or enter a new name to add a new parameter. Select an existing Test environment variable name from the **Name** box to modify an existing parameter.

Enter the value as a regular expression in the **Value** box.

Note: If you select an external user-defined or built-in environment variable name, the Value box and the Regular Expression check box are disabled. The variable type of the selected parameter is displayed below the Value box.

For information on regular expression syntax, see "Regular Expression Syntax," on page 218. For information on parameterization, see Chapter 10, "Parameterizing Tests."

---

4 Select the **Regular Expression** check box.

5 For environment variable parameters, you are prompted to add a backslash (\) before each special character in the **Value** text box if you want to treat it literally.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), hyphen (-), asterisk (*), caret (^), brackets ([ ]), parentheses (()), dollar sign ($), vertical line (|), plus sign (+), question mark (?), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as literal character.

- Click **Yes** to instruct QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
- Click **No** to instruct QuickTest to treat all the characters literally, except for special characters.

6 Click **OK** to close the Parameter Options dialog box.

7 Click **OK** to save and close the Object Properties dialog box.

In your test tree, the icon next to the step indicates that the step has been parameterized.

Using Regular Expressions in Object Checkpoints

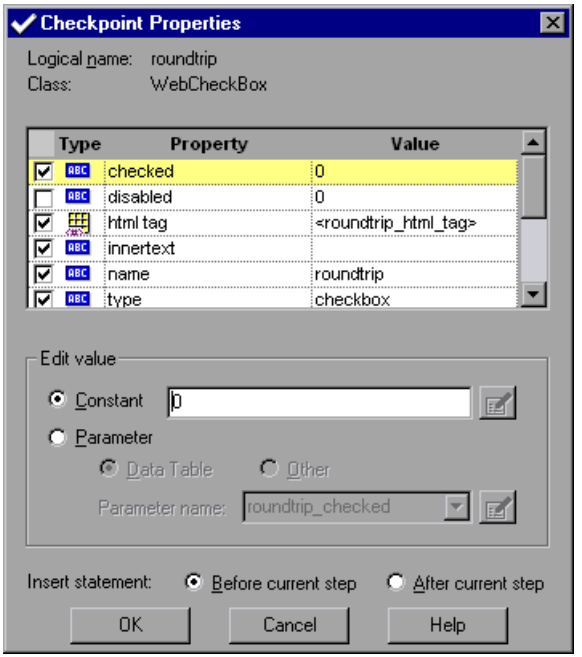
When creating an object checkpoint to verify that an object is displayed on your Web site, you can also set the property value of the object as a regular expression, so that an object with a varying name can be verified.

For example, suppose you want to check that when booking the number of passengers for a flight reservation in the "Mercury Tours" Web site, whole numbers are used. QuickTest will ignore variations in the object's property value as long as the value is a whole number.

To define a regular expression in an object checkpoint:

1 Right-click the object in the ActiveScreen and choose **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.

2 Select the object in the **Object Selection - Checkpoint Properties** dialog box and click **OK**. The Checkpoint Properties dialog box opens.



The Checkpoint Properties dialog box enables you to specify which properties of the object to check, and to edit the values of these properties. For more information, see Chapter 6, “Creating Checkpoints.”

3 In the **Property** section, click the property you want to set as a regular expression. The property is highlighted.

4 In the **Edit value** section, click **Constant**.

5 Click the **Edit Constant Value Options** button. The Constant Value Options dialog box opens.

In the **Value** box, enter the regular expression syntax for the string. For information on regular expression syntax, see "Regular Expression Syntax," on page 218.

6 Select the **Regular Expression** check box.

7 You are prompted to add a backslash (\) before each special character in the **Constant** text box if you want to treat it literally.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), brackets ([ ]), hyphen (-), caret (^), asterisk (*), plus sign (+), question mark (?), parentheses (()), dollar sign ($), vertical line (|), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as literal character.

- Click **Yes** to instruct QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
- Click **No** to instruct QuickTest to treat all the characters literally, except for special characters.

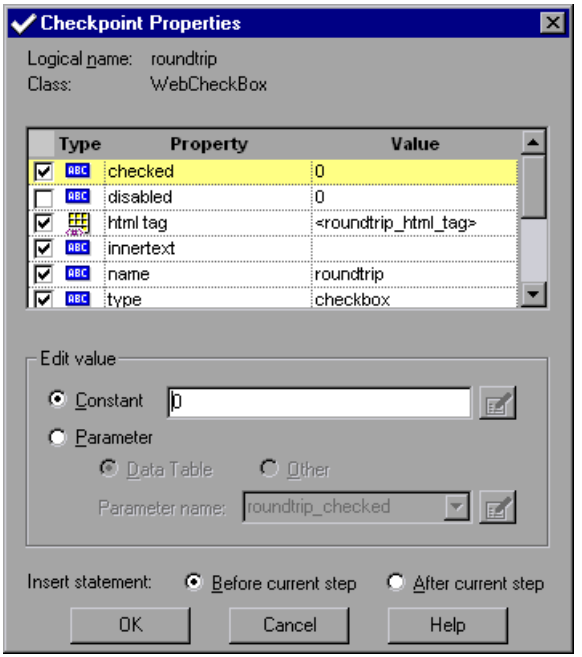
8 Click **OK** to close the Constant Value Options dialog box.

9 Click **OK** to save and close the Checkpoint Properties dialog box.

To parameterize an object checkpoint using a regular expression:

1 Right-click the object in the ActiveScreen and choose **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.

2 Select the object in the **Object Selection - Checkpoint Properties** dialog box and click **OK**. The Checkpoint Properties dialog box opens.



3 In the **Property** section, click the property you want to set as a regular expression. The property is highlighted.

4 In the **Edit value** section, click **Parameter**. Specify the type of parameter you want:

➤ Click **Data Table** to create a data table parameter. Click the **Edit Parameter Options** button to open the **DataTable Parameter Options** dialog box.

In the **Name** box, choose a parameter from the list or enter a new name. To use a parameter that you already created, select it from the list. To create a new parameter, either use the default parameter name or enter a

descriptive name for the parameter. For more information on parameterization, see Chapter 10, "Parameterizing Tests."

To add the parameter to the Global tab in the Data pane, select **Global sheet**. To add the parameter to the Action tab, select **Current action sheet (local)**. For more information about Actions, see Chapter 14, "Working with Actions."

Note: The value in the **Name** box should not be defined as a regular expression. When you add additional values for the parameter into the table in the Data pane, you specify the values as regular expressions.

For information on regular expression syntax, see "Regular Expression Syntax," on page 218. For information on parameterization, see Chapter 10, "Parameterizing Tests."

➤ Click **Other** to create a parameter other than a data table parameter. Click the **Edit Parameter Options** button to open the **Parameter Options** dialog box.

Choose **Environment**. Accept the default name or enter a new name to add a new parameter. Select an existing Test environment variable name from the **Name** box to modify an existing parameter.

Enter the value as a regular expression in the **Value** box.

Note: If you select an external user-defined or built-in environment variable name, the Value box and the Regular Expression check box are disabled. The variable type of the selected parameter is displayed below the Value box.

For information on regular expression syntax, see "Regular Expression Syntax," on page 218. For information on parameterization, see Chapter 10, "Parameterizing Tests."

---

5 Select the **Regular Expression** check box.

6 For environment variable parameters, you are prompted to add a backslash (\) before each special character in the **Value** text box if you want to treat it literally.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), hyphen (-), asterisk (*), caret (^), brackets ([ ]), parentheses (()), dollar sign ($), vertical line (|), plus sign (+), question mark (?), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as literal character.

- Click **Yes** to instruct QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
- Click **No** to instruct QuickTest to treat all the characters literally, except for special characters.

7 Click **OK** to close the Parameter Options dialog box.

8 Click **OK** to save and close the Checkpoint Properties dialog box.

In your test tree, the  icon next to the step indicates that the step has been parameterized. The  icon indicates a checkpoint.

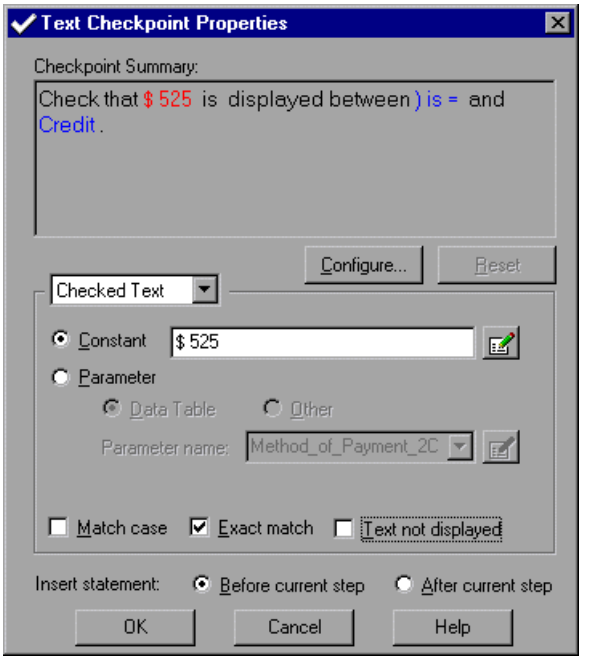
Using Regular Expressions in Text Checkpoints

When creating a text checkpoint to check that a varying text string is displayed on your Web site, you define the text string as a regular expression.

For example, when booking a flight in the “Mercury Tours” sample site, the total cost charged to a credit card number should not be less than $300. You define the amount as a regular expression, so that QuickTest will ignore variations in the text string as long as the value is not less than $300.

To define a regular expression in a text checkpoint:

1 Highlight the text you want to check in the ActiveScreen. Right-click the selected text and choose **Insert Text Checkpoint**. The Text Checkpoint Properties dialog box opens.



The Text Checkpoint Properties dialog box enables you to specify which text to check as well as which text is displayed before and after the text to check. For more information, see Chapter 6, "Creating Checkpoints."

2 In the **Checked Text** section, click **Constant**.

3 Click the **Edit Constant Value Options** button. The Constant Value Options dialog box opens.

In the **Value** box, enter the regular expression syntax for the string. For information on regular expression syntax, see "Regular Expression Syntax," on page 218.

4 Select the **Regular Expression** check box.

5 You are prompted to add a backslash (\) before each special character in the **Constant** text box if you want to treat it literally.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), brackets ([ ]), hyphen (-), caret (^), asterisk (*), plus sign (+), question mark (?), parentheses (()), dollar sign ($), vertical line (|), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as literal character.

- Click **Yes** to instruct QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
- Click **No** to instruct QuickTest to treat all the characters literally, except for special characters.

6 Click **OK** to close the Constant Value Options dialog box.

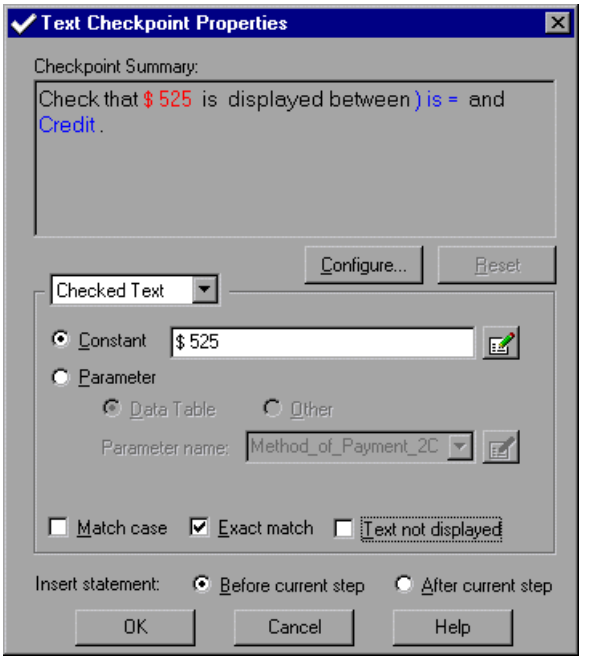
7 Select to check the text by Match case, Exact match, and/or Text not displayed. For more information on these options, see "Specifying the Checked Text," on page 94.

8 If you want to check a specific occurrence of the selected text, define the text displayed before and after the checked text. For more information, see "Specifying the Text Displayed Before the Checked Text," on page 96, and "Specifying the Text Displayed After the Checked Text," on page 99.

9 Click **OK** to save and close the Text Checkpoint Properties dialog box.

To parameterize a text checkpoint using a regular expression:

1 Highlight the text you want to check in the ActiveScreen. Right-click the selected text and choose **Insert Text Checkpoint**. The Text Checkpoint Properties dialog box opens.



2 In the **Checked Text** section, click **Parameter**. Specify the type of parameter you want:

➤ Click **Data Table** to create a data table parameter. Click the **Edit Parameter Options** button to open the **DataTable Parameter Options** dialog box.

In the **Name** box, choose a parameter from the list or enter a new name. To use a parameter that you already created, select it from the list. To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter. For more information on parameterization, see Chapter 10, "Parameterizing Tests."

To add the parameter to the Global tab in the Data pane, select **Global sheet**. To add the parameter to the Action tab, select **Current action sheet (local)**. For more information about Actions, see Chapter 14, "Working with Actions."

Note: The value in the **Name** box should not be defined as a regular expression. When you add additional values for the parameter into the table in the Data pane, you specify the values as regular expressions.

For information on regular expression syntax, see "Regular Expression Syntax," on page 218. For information on parameterization, see Chapter 10, "Parameterizing Tests."

➤ Click **Other** to create a parameter other than a data table parameter. Click the **Edit Parameter Options** button to open the **Parameter Options** dialog box.

Choose **Environment**. Accept the default name or enter a new name to add a new parameter. Select an existing Test environment variable name from the **Name** box to modify an existing parameter.

Enter the value as a regular expression in the **Value** box.

Note: If you select an external user-defined or built-in environment variable name, the Value box and the Regular Expression check box are disabled. The variable type of the selected parameter is displayed below the Value box.

For information on regular expression syntax, see "Regular Expression Syntax," on page 218. For information on parameterization, see Chapter 10, "Parameterizing Tests."

---

3 Select the **Regular Expression** check box.

4 For environment variable parameters, you are prompted to add a backslash (\) before each special character in the **Value** text box if you want to treat it literally.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), hyphen (-), asterisk (*), caret (^), brackets ([ ]), parentheses (()), dollar sign ($), vertical line (|), plus sign (+), question mark (?), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as literal character.

- Click **Yes** to instruct QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
- Click **No** to instruct QuickTest to treat all the characters literally, except for special characters.

5 Click **OK** to close the Parameter Options dialog box.

6 If you want to check a specific occurrence of the selected text, define the text displayed before and after the checked text. For more information, see "Specifying the Text Displayed Before the Checked Text," on page 96, and "Specifying the Text Displayed After the Checked Text," on page 99.

7 Click **OK** to save and close the Text Checkpoint Properties dialog box.

In your test tree, the  icon next to the step indicates that the step has been parameterized. The  icon indicates a checkpoint.

Regular Expression Syntax

QuickTest searches for all characters in a regular expression literally, except for the period (.), brackets ([ ]), list brackets ([:...:]), curly brackets ({...}), hyphen (-), caret (^), asterisk (*), plus sign (+), question mark (?), parentheses (()), dollar sign ($), vertical line (|), and backslash (\) as described below. When one of these special characters is preceded by a backslash (\), QuickTest searches for the literal character.

The following options can be used to create regular expressions:

Matching Any Single Character

A period (.) instructs QuickTest to search for any single character. For example:

```
welcome.
```

matches welcomes, welcomed, or welcome followed by a space or any other single character. A series of periods indicates the same number of unspecified characters.

Matching Any Single Character within a Range

In order to match a single character within a range, you can use square brackets ([ ]). For example, to search for a date that is either 1968 or 1969, write:

```
196[89]
```

You can use a hyphen (-) to indicate an actual range. For instance, to match any year in the 1960s, write:

```
196[0-9]
```

A hyphen does not signify a range if it is displayed as the first or last character within brackets, or after a caret (^).

A caret (^) instructs QuickTest to match any character except for the ones specified in the string. For example:

```
[^A-Za-z]
```

matches any non-alphabetic character. The caret has this special meaning only when it is displayed first within the brackets.

Note that within brackets, the characters ".", "*", "[" and "\" are literal. If the right bracket is the first character in the range, it is also literal. For example:

```
[]g-m]
```

matches the right bracket, and g through m.

Matching Zero or More Specific Characters

An asterisk (*) instructs QuickTest to match zero or more occurrences of the preceding character. For example:

```
ca*r
```

matches car, caaaaaar, and cr.

Matching One or More Specific Characters

A plus sign (+) instructs QuickTest to match one or more occurrences of the preceding character. For example:

```
ca+r
```

matches car and caaaaaar, but not cr.

Matching Zero or One Specific Characters

A question mark (?) instructs QuickTest to match zero or one occurrences of the preceding character. For example:

```
ca?r
```

matches car and cr, but nothing else.

Grouping Regular Expressions

Parentheses (()) instruct QuickTest to treat the contained sequence as a unit, just as in mathematics and programming languages.

Matching one of several regular expressions

A vertical line (|) instructs QuickTest to match one of a choice of expressions. For example:

```
foo|bar
```

causes QuickTest to match either foo or bar.

```
fo(o|b)ar
```

causes QuickTest to match either fooar or fobar.

Combining Special Characters

You can combine special characters in a regular expression.

For example you can combine the '.' and '*' characters in order to find zero or more occurrences of any character.

For example,

```
start.*
```

matches start, started, starting, starter, etc.

You can also use a combination of brackets and an asterisk to limit the search to a combination of non-numeric characters. For example:

```
"[a-zA-Z]*"
```

Matching the End of a String

A dollar sign ($) instructs QuickTest to match the end of a string. For example:

```
book.*
```

matches both book and bookend, while a string that is followed by ($), matches only that string. For example,

```
book$
```

matches only book.

Using the Backslash Character

A backslash (\) instructs QuickTest to treat the next character as either a special character or a literal character. For example:

- n matches the character n
- \n matches a newline character
- \\ matches \
- \(matches (

If you were looking for a Web site called

mercurytours.mercuryinteractive.com,

the period would be mistaken as an indication of a regular expression. To indicate that the period is not part of a regular expression, you would write it as follows:

mercurytours\.mercuryinteractive\.com.

For more information about regular expression syntax, refer to: http://sunland.gsfc.nasa.gov/info/regex/Top.html

Matching character class operators

Brackets surrounding opening and closing colons ([:...:]) instruct QuickTest to search for character class expressions inside lists. For example:

[[:alpha:]]
matches any letter and:

[[:digit:]]
matches any digit.

Therefore: a[[:digit:]] is the same as: a[1-9]. It matches a1, a2, a3, etc.

Matching interval operators.

Curly brackets ({...}) instruct QuickTest to match the number of occurrences of a given operator in the following manner:

{*count*} matches the exact number of occurrences of the preceding regular expression.

{*min,* } indicates the minimum number of occurrences of the preceding regular expression.

{*min, max*} indicates the minimum and maximum number of occurrences of the preceding regular expression.

For example:

(a(b))\2{3} matches abbb

Matching a Digit Character

You can use the character "\d" to instruct QuickTest to match any digit character (0-9).

For example:

\d* causes QuickTest to match zero or more occurrences of the digits 0-9. It matches 21, 4786, or 87450683.

For example:

1\d* causes QuickTest to match zero or more occurrences of the digits 0-9 preceded by the digit 1. It matches 1, 134, or 187450683.

Matching any Word Character Including the Underscore

You can use the character "\w" to instruct QuickTest to match any word character and the underscore (A-Z, a-z, 0-9, _).

For example:

\w* causes QuickTest to match zero or more occurrences of the word characters: A-Z, a-z, 0-9, and the underscore (_). It matches Ab, r9Cj, or 12_uYLgeu_435.

For example:

\w{3} causes QuickTest to match 3 occurrences of the word characters A-Z, a-z, 0-9, and the underscore (_). It matches Ab4, r9_, or z_M.

For more information about using regular expressions, and for information on additional regular expression options such as (\b) and (\B), refer to the Microsoft HTML Help by choosing **Help > Microsoft VBScript Reference.** Click the **Index** tab, browse to **regular expression syntax**, and click **Display.**

13

Learning Virtual Objects

You can teach QuickTest to recognize any area of your Web-based application as an object by defining it as a *virtual object*. Virtual objects enable you to run tests on objects that are not recognized by QuickTest and obtain useful test results.

This chapter describes:

- Understanding Virtual Objects
- Defining a Virtual Object
- Removing a Virtual Object

About Learning Virtual Objects

Your Web-based application may contain objects that behave like standard objects but are not recognized by QuickTest. You can define these objects as virtual objects and map them to standard classes, such as a button or a check box. QuickTest emulates the user's action on the virtual object during the test run. In the test results, the virtual object is displayed as though it is a standard class object.

For example, suppose you want to record a test on a Web page containing a bitmap that the user clicks. The bitmap contains several different hyperlink areas, and each area opens a different destination page. When you record a test, the Web site matches the coordinates of the click on the bitmap and opens the destination page. To enable QuickTest to click at the required coordinates during a test run, you can define a virtual object for an area of the bitmap, which includes those coordinates, and map it to the button class. When you run a test, QuickTest clicks the bitmap in the area defined as a virtual object so that the Web site opens the correct destination page.

You define a virtual object using the Virtual Object wizard. The wizard prompts you to select the standard object class to which you want to map the virtual object. You then mark the boundaries of the virtual object using a crosshairs pointer. Next, you select a test object as the parent of the virtual object. Finally, you specify a name and a *collection* for the virtual object. QuickTest adds the virtual object to the collection, which is a group of virtual objects with a descriptive name that is stored in the Virtual Object Manager.

Note: QuickTest does not support virtual objects for low-level recording. For additional information about low-level recording, see "Recording and Running Tests," on page 377.

Understanding Virtual Objects

QuickTest identifies a virtual object according to its boundaries. Marking an object's boundaries specifies its size and position on a Web page. When you assign a test object as the parent of your virtual object, you specify that the coordinates of the virtual object boundaries are relative to that parent object. When you record a test, QuickTest recognizes the virtual object within the parent object, and adds it as a test object in the Object Repository so that QuickTest can identify the object during the test run. Note that the Web page must be the same size and in the same position when recording and running tests as it was when you defined the virtual object.

You can disable recognition of virtual objects without deleting them from the Virtual Object Manager. For additional information, see "Removing a Virtual Object," on page 231.

Note: You can define virtual objects only for objects on which you can *click* or *double-click* and which record a **Click** or **DblClick** step in the test tree. Otherwise, the virtual object is ignored. For example, if you define a virtual object over the WinList object, the **Select** operation is recorded, and the virtual object is ignored.

Defining a Virtual Object

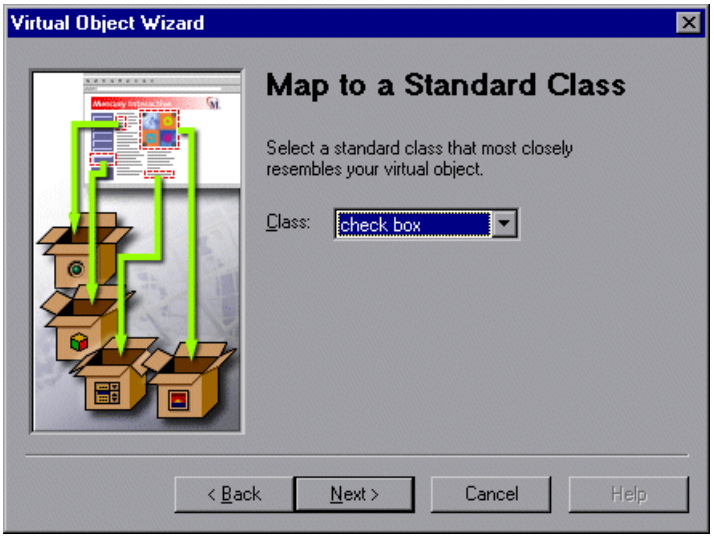
Using the Virtual Object wizard, you can map a virtual object to a standard object class, specify the boundaries and the parent of the virtual object, and assign it a logical name. You can also group your virtual objects logically by assigning them to collections.

To define a virtual object:

1 With QuickTest open (but not in record mode), open your Web browser to the Web page containing the area you want to define as a virtual object.

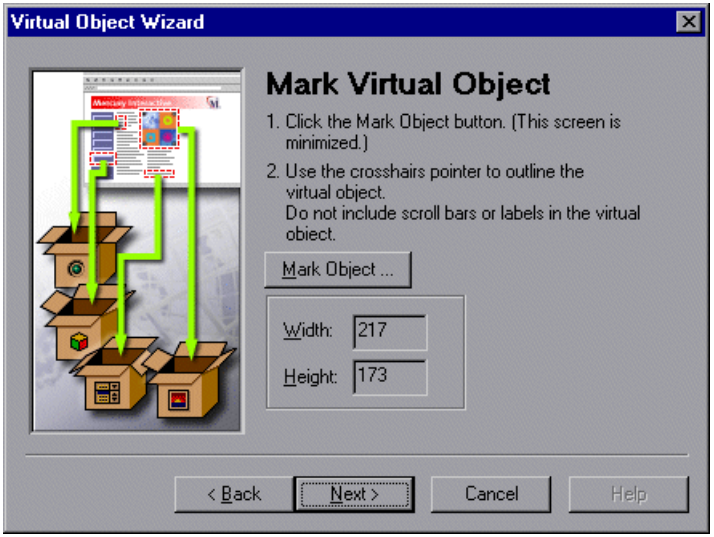
2 In QuickTest, choose **Tools** > **Virtual Objects** > **New Virtual Object**. The Virtual Object wizard opens. Click **Next**.

3 Select a standard class to which you want to map your virtual object.



If you select the list class, specify the number of rows in the virtual object. For the table class, select the number of rows and columns. Click **Next**.

4 Click **Mark Object**.

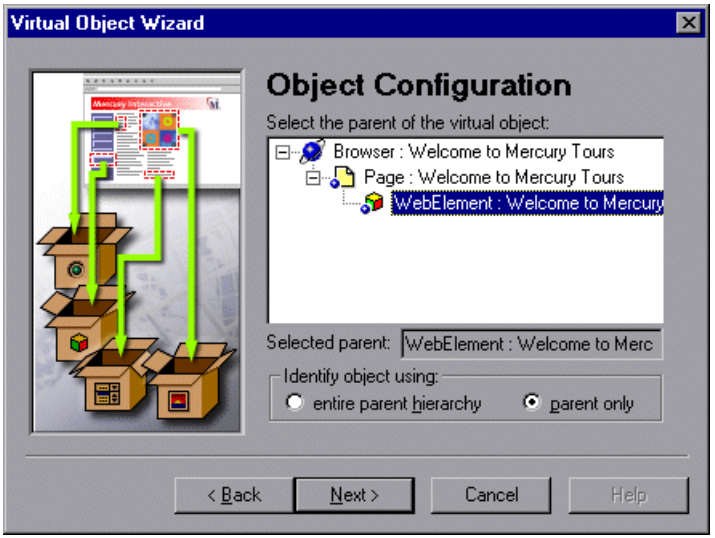


The QuickTest window and the Virtual Object wizard are minimized. Use the crosshairs pointer to mark the area of the virtual object. You can use the arrow keys while holding down the left mouse button to make precise adjustments to the area you define with the crosshairs. Click **Next**.

Note: The virtual object should not overlap other objects in your Web page. If the virtual object overlaps another object, QuickTest may not record or run tests properly on the object and/or the virtual object.

---

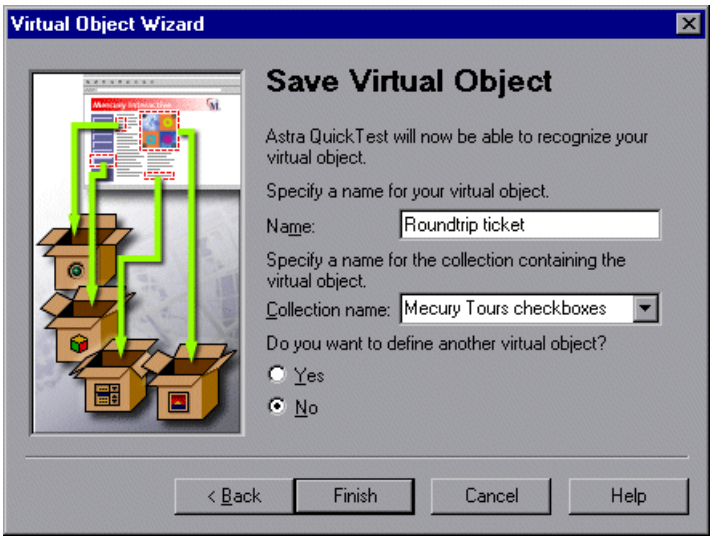
5 Click an object in the object tree to assign it as the parent of the virtual object.



The coordinates of the virtual object outline are relative to the parent object you select.

6 In the **Identify object using** box, select how to display the virtual object in the test tree. Click **Next**.

7 Specify a name and a collection for the virtual object. Choose from the list of collections or create a new one by entering a new name in the **Collection name** box.



8 To add the virtual object to the Virtual Object Manager and close the wizard, select **No** and then click **Finish**.

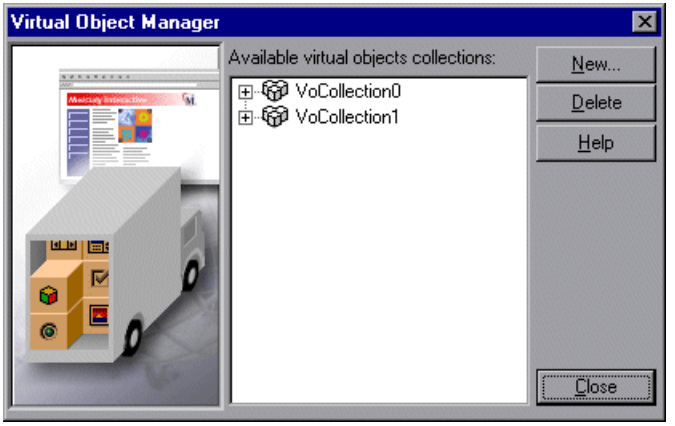
To add the virtual object to the Virtual Object Manager and define another virtual object, select **Yes** and then click **Next**. The wizard returns to the Map to a Standard Class screen, where you can define the next virtual object.

Removing a Virtual Object

You can remove virtual objects from your test by deleting them or by disabling recognition of these objects while recording.

To delete a virtual object:

1 Choose **Tools** > **Virtual Objects** > **Virtual Object Manager.** The Virtual Object Manager opens.



2 In the list of available virtual object collections, click the plus sign next to the collection to display the virtual object you want to delete. Select the virtual object, and click **Delete**.

To delete an entire collection, select it, and click **Delete**.

3 Click **Close**.

Tip: To define a new virtual object, click **New** in the Virtual Object Manager to open the Virtual Object wizard.

To disable recognition of virtual objects while recording:

1. Choose **Tools** > **Options**. The Options dialog box opens.
2. In the General tab, select the **Disable recognition of virtual objects when recording** check box.
3. Click **OK**.

Note: When you want QuickTest to recognize virtual objects during recording, ensure that the **Disable recognition of virtual objects when recording** check box in the General tab of the Options dialog box is cleared. For more information, see "General Testing Options," on page 387.

14

Working with Actions

You can divide your test into actions in order to streamline the testing process of your Web sites.

This chapter describes:

- Using Multiple Actions in a Test
- Parameterizing Actions
- Using the Action Toolbar
- Creating New Actions
- Inserting Existing Actions
- Nesting Actions
- Splitting Actions
- Setting Action Properties
- Exiting an Action
- Removing Actions From a Test
- Guidelines for Working with Actions

About Working with Actions

Actions divide your test into logical sections, like the main sections of a Web site. When you create a new test, it contains one action. By dividing your tests into multiple actions, you can design more modular and efficient tests.

Actions enable you to parameterize specific components of a test and to easily re-record one action so that you don't have to re-record the entire test when part of your application changes.

Two or more tests can *call* (link to) the same action and one action can call (link to) another action. Complex tests may have many actions, and may share actions with other tests.

Using Multiple Actions in a Test

When you create a test, it includes one action. All the steps you record and all the modifications you make after recording are part of a single action.

You can divide your test into multiple actions by creating new actions or by inserting existing actions. There are three kinds of actions:

- **non-reusable action** - an action that can only be used in the test in which it was created, and only once.
- **reusable action** - an action that can be called multiple times by the test in which it was created as well as by other tests.
- **external action** - a reusable action created in another test. External actions are read-only in the calling test. They can only be modified from the original test.

An existing action can be inserted as a copy of the original action, as a call to a reusable action from the local test or a call to an external action. For more information about creating new actions, see "Creating New Actions," on page 238. For more information about inserting existing actions see "Inserting Existing Actions," on page 240.

By default, new actions are non-reusable. You can mark each action you create in a test as reusable or non-reusable. Only reusable actions can be called from the current test or from another test.

For every action in your test, QuickTest creates a corresponding *current action* sheet in the Data pane so that you can enter data table parameters that are specific to that action only. For more information on parameterizing actions, see "Parameterizing Actions," on page 235. For information on parameterizing tests, see Chapter 10, "Parameterizing Tests," and Chapter 11, "Creating Output Values."

When you run a test with actions, the Test Results are divided by actions within each test iteration so that you can see the outcome of each action, and you can view the detailed results for each action individually. For more information on the Test Results window, see Chapter 18, "Analyzing Test Results."

Note: You can also run a single action, or part of an action, using the Run Action from Step option. For more information, see Chapter 17, "Running Tests."

Parameterizing Actions

You parameterize an action by parameterizing the steps within that action. When you add a data table parameter to your test, you specify whether it is a *global* data table parameter, for the entire test, or a *current action* data table parameter, for a specific action within the test.

In the data table parameter options dialog boxes:

- Choosing **Global sheet** creates parameters in the **Global** sheet in the Data pane.
- Each action has its own sheet in the data table so that you can insert data that applies only to that action. Choosing **Current action sheet (local)** creates parameters in the corresponding action sheet in the Data pane. When there are parameters in a current action's sheet, you can set QuickTest to run one or more iterations on that action before continuing with the test. For more information about setting action iteration preferences, see "Setting the Run Properties for an Action," on page 254.

Note: If you create data table parameters in your action, be sure that the run settings for your action are set correctly in the Run tab of the Action Properties dialog box. You can set your action to run without iterations, to run iterations on all rows in the action's data sheet, or to run iterations only on the rows you specify. For more information about the Run tab settings, see "Setting the Run Properties for an Action," on page 254.

For more information on parameterizing, see Chapter 10, "Parameterizing Tests."

For more information on the data table, see Chapter 15, "Working with Data Tables."

When you create a data table parameter, the original constant value is entered as the first row in the parameter column. You can specify additional data values for the parameter by entering them directly into the relevant sheet in the Data pane. For more information, see Chapter 15, "Working with Data Tables."

For more information about parameterization, see Chapter 10, "Parameterizing Tests."

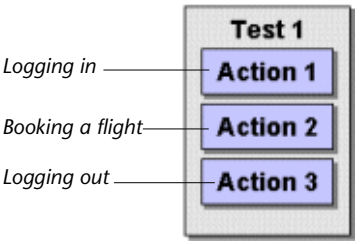
Suppose you want to test how a flight reservation system handles multiple bookings. You may want to parameterize the test to check how your site responds to multiple sets of customer flight itineraries. When you plan your test, you plan the following procedures:

1. The travel agent logs into the flight reservation system.
2. The travel agent books five sets of customer flight itineraries.
3. The travel agent logs out of the flight reservation site.

When you consider these procedures, you realize that it is necessary to parameterize only the second step: after all, the travel agent logs into the flight reservation system only once, at the beginning and logs out of the system only once, at the end. Therefore, it is not necessary to parameterize the login and logout procedures in your test.

By creating three separate actions within your test—one for logging in, another for booking a flight, and a third for logging out—you can parameterize the second action in your test without parameterizing the others.

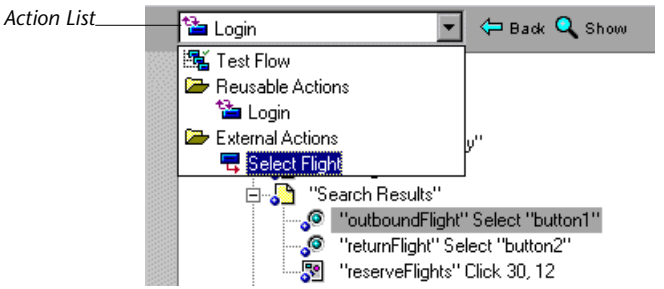
When you divide your test into three actions, it is structured as shown:



Using the Action Toolbar

By default, QuickTest opens with the Action toolbar hidden in the Tree View. You can choose to view it at any time by enabling the **Display Action toolbar in Tree View** option from the General tab of the Options dialog box. For more information, see Chapter 24, "Setting Global Testing Options."

The first time you insert a reusable or external action in a test, the **Display Action toolbar in Tree View** option is automatically enabled and the Action toolbar is displayed above the test tree.

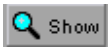


The *Action List* enables you to view either the entire test flow or an action view for a selected reusable or external action. The *test flow* displays the overall flow of your test with all the actions in your test. An *action view* displays all the details of the selected reusable or external action.

In the test flow, reusable and external actions are not expandable. You can view the expanded tree of reusable and external actions by opening the action view for that action. For more information about reusable actions, see "Setting the Reusability Status of an Action," on page 252.

There are three ways to switch to the action view for a reusable or external action:

- Double-click the action you want to view.
- Highlight the action you want to view and click the Show button.



- Select the name of the action from the Action List.

Note: You can open and expand the view for an external action, but the action is read-only. The action can only be modified in the original test. For more information about external actions, see "Inserting a Call to an Action," on page 243.

If you are working in a test without reusable or external actions, you can hide the Action toolbar. To hide the Action toolbar, clear the **Display Action toolbar in the Tree View** option in the General Tab of the Options dialog box. For more information, see Chapter 24, "Setting Global Testing Options."

In the Expert view, the Action toolbar is always visible, and the Expert view always displays the script for the selected action. For more information on the Expert View, see Chapter 22, "Testing in the Expert View."

Creating New Actions

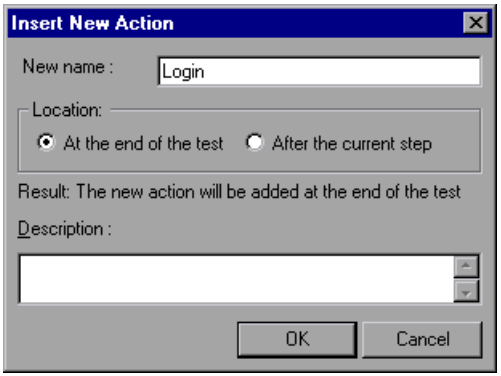
You can add new actions to your test during a recording session or before you begin a recording session.

You can also split an existing action into two actions. For more information on splitting actions, see "Splitting Actions," on page 248.

To create a new action in your test:

1 If you want to insert the action within an existing action, click the step after which you want to insert the new action.

2 Choose **Insert** > **New Action** or click the **New Action** button. The Insert New Action dialog box opens.



3 Type a new action name or accept the default name.

4 Decide where to insert the action:

- To insert a new action at the end of your test, select **At the end of the test**.
- To insert the new action within the action of the currently selected step, select **After the current step**.

For more information about inserting actions within actions, see "Nesting Actions," on page 246.

5 If you wish, add a description of the action. You can also add an action description at a later time in the Action Properties dialog box.

Tip: Descriptions of actions are displayed in the Insert Copy of Action and Insert Call to Action dialog boxes. The description makes it easier for you to select the action you want to insert. For more information, see "Adding or Editing an Action Description," on page 251.

---

6 Click **OK**. A new action is added to your test and is displayed at the bottom of the test tree or after the current step.

7 Make sure your new action is selected and click **Record** to continue recording. The steps you record will be added to your new action.

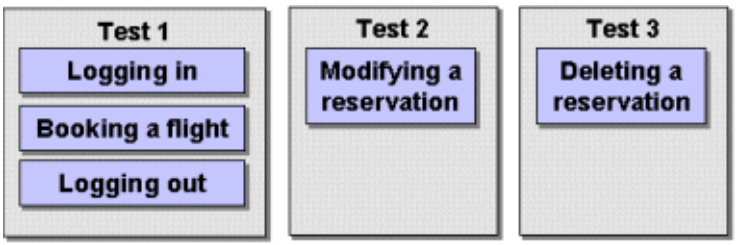
By default, all new actions are inserted as non-reusable actions. If you want to be able to call the action from within your test or from other tests, you can make it a reusable action. For more information about reusable actions, see "Setting the Reusability Status of an Action," on page 252.

Inserting Existing Actions

When you plan a suite of tests, you may realize that each test requires one or more identical activities, such as logging in. For example, rather than recording the login process three times in three separate tests, and enhancing this part of the script (with checkpoints and parameterization) separately for each test, you can create an action that logs into the flight reservation system in one test. Once you are satisfied with the action you recorded and enhanced, you can insert the existing action into other tests.

You can insert an existing action by inserting a copy of the action into your test, or by inserting a call to the original action.

Suppose you wanted to record the following three tests in the Mercury Tours site: Booking a flight, Modifying a reservation and Deleting a reservation. While planning your test you realize that for each test you need to log in and log out of the site. If you plan to use the insert existing action option for the repeated activities, then you would initially record the three tests as shown:



Once you have set up your tests, you must choose whether you want to insert a copy of the action or insert a call to it.

About Reusable Actions

A reusable action can be called multiple times within a test, and can be called from other tests. Non-reusable actions can be copied and inserted as independent actions, but cannot be inserted as calls to the original action.

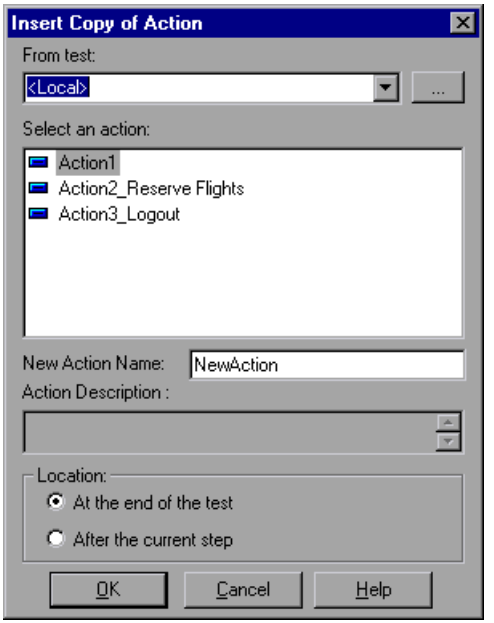
Inserting calls to reusable actions makes it easier to maintain your tests because when an object or procedure in your application changes, it only needs to be updated one time, in the original action.

Inserting a Copy of an Action

When you insert a copy of an action into a test, the action is copied in its entirety, including checkpoints, parameterization, and the corresponding action tab in the Data pane. The action is inserted into the test as an independent, non-reusable action (even if the original action was reusable). Once the action is copied into your test, you can add to, delete from, or modify the action just as you would with any other recorded action. Any changes you make to this action after you insert it affect only this action, and changes you make to the original action do not affect the inserted action. You can insert copies of both reusable and non-reusable actions.

To insert a copy of an action:

1 Choose **Insert** > **Copy of Action** or right-click the action and select **Copy Action**. The Insert Copy of Action dialog box opens.



2 Use the browse button to find the test from which you want to insert the action. The action window displays all *local* actions (actions that originated) in the test you selected.

3 Select the action you want to insert from the list. When you highlight an action, its description, if one exists, is displayed in the Action Description box. This helps you identify the action you want to insert. For more information about action descriptions see "To add or edit an action description:," on page 252.

4 Type a meaningful name for the action in the New Action Name box.

5 Decide where to insert the action:

- To insert a new action at the end of your test, select **At the end of the test**.
- To insert the new action within the action of the currently selected step, select **After the current step**.

For more information about inserting actions within actions, see "Nesting Actions," on page 246.

6 Click **OK**. The action is inserted into the test as an independent, non-reusable action. You can move the action to another location in the test by dragging it to the desired location in the test tree.

Inserting a Call to an Action

You can insert a call (link) to a reusable action that resides in your current test (local action), or in any other test (external action).

When you insert a call to an external action, the action is inserted in read-only format. You can view the components of the action in the action view, but you cannot modify them.

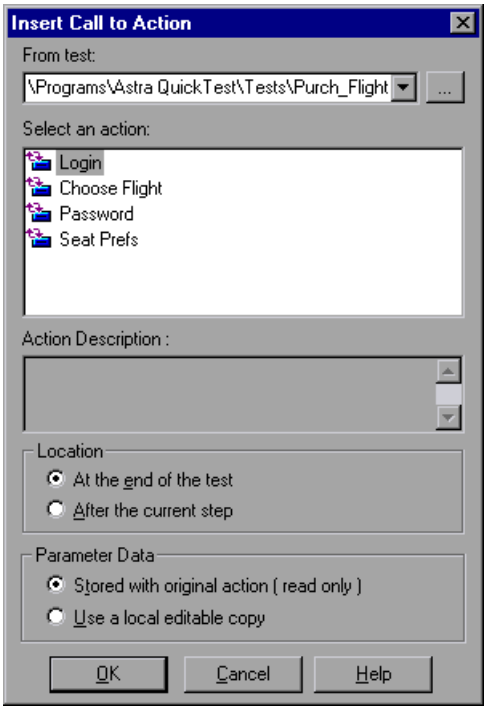
You can choose, however, whether you want the data from the action's data sheet to be imported as a local, editable copy, or whether you want to use the (read-only) data from the original action.

To modify the action, you must open the test in which the action originated. In the original action, the modifications apply to all tests that call that action. If you chose to use the original action's data, then changes to the original action's data will be applied as well.

Tip: You can view the location of the original action in the General tab of the Action Properties dialog box.

To insert a call to an action:

1 Choose **Insert** > **Call to Action** or right-click the action and select **Call Action**. The Insert Action dialog box opens.



2 Use the browse button to find the test from which you want to insert the action. The action window displays all reusable and external actions in the test you selected.

3 Select the action you want to insert from the list. When you highlight an action, its description, if one exists, is displayed in the Action Description box. This helps you identify the action you want to insert. For more information about action descriptions see "To add or edit an action description:," on page 252.

4 Choose where to insert the action:

- To insert a new action at the end of your test, select **At the end of the test**.
- To insert the new action within the action of the currently selected step, select **After the current step**.

For more information about nesting actions, see "Nesting Actions," on page 246.

5 Choose where you want to store the parameterization data:

- To use the original action's data, select **Use data stored with the original action (read-only)**. When you select this option, the data is read-only when viewed from the calling test and all changes to the original action's data sheet apply when the action runs in the calling test.
- To store an editable copy of the data in the test's data table, select **Use a local, editable copy**. When you select this option, the data sheet is stored in the test's data table and is independent of the original action. Changes to the original data sheet do not affect the calling test.

Note: This option is only enabled the first time you insert a call to a given action and the option setting applies to all calls to the same action within the test. You can change the parameterization storage location for the action in the Action Properties dialog box. For more information, see "Setting Action Properties," on page 250.

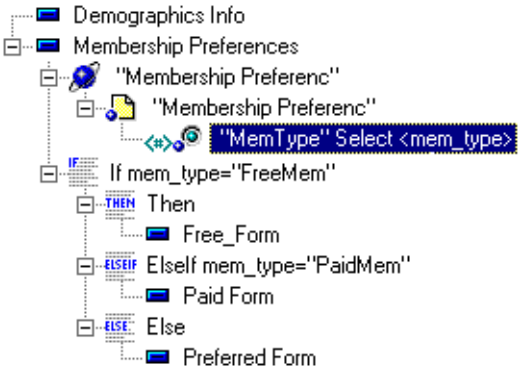
6 Click **OK**. The action is inserted into the test as a call to the original action. You can move the action to another location in the test by dragging it to the desired location in the test tree.

Nesting Actions

Sometimes you may want to run an action within an action. This is called *nesting*. Nesting actions can:

- help you maintain the modularity of your test
- enable you to run one action or another based on the results of a conditional statement

For example, suppose you have parameterized a step where a user selects one of three membership types as part of a registration process. When the user clicks **Continue** on the registration form, the page that opens depends on the membership type selected in the previous page. You can create one action for each type of membership. Then you can use *if* statements to determine which membership type was selected in a particular iteration of the test, and run the appropriate action for that selection. Your script might look something like this:



For more information about inserting conditional statements, see "Using Conditional Statements," on page 349.

To nest an action within an existing action:

1. Highlight the step after which you would like to insert the action.
2. Follow the instructions for creating a new action as described on "Creating New Actions," on page 238, or follow the instructions for inserting an existing action as described on "Inserting Existing Actions," on page 240.

3 In the Location section of the Insert New Action, Insert Copy of Action or Insert Call to Action dialog box, select **After the current step**. The action is inserted after the current step.

Note: In the Expert View, an action called by another action is displayed within the parent action with the following syntax:

Call RunAction (*ActionName*, *Run_Setting*, *FromRow* - *ToRow* **)**

For example:

Call RunAction("Select Flight", 0, "1 - 1")

For more information about the Expert View, see Chapter 22, “Testing in the Expert View.”

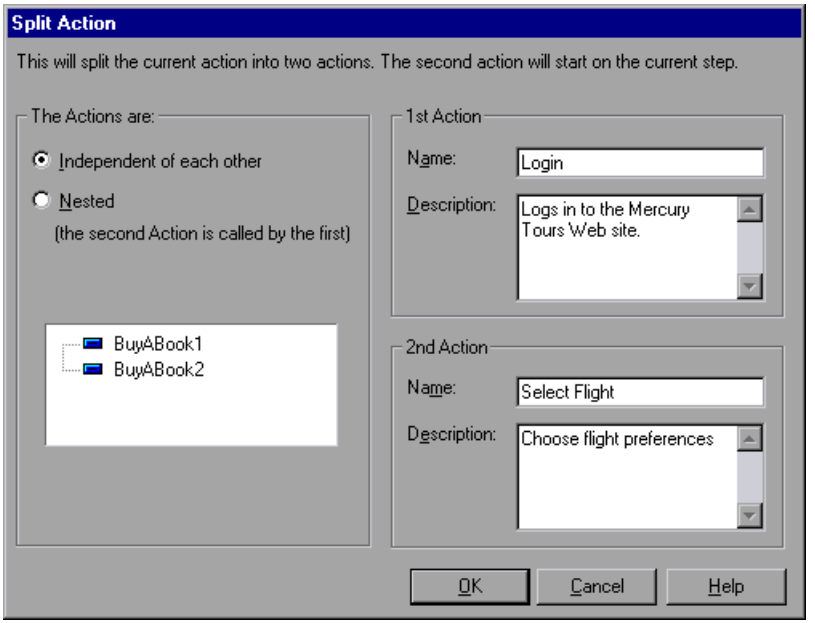
Splitting Actions

You can split an existing action into two sibling actions or into parent-child nested actions.

To split an action:

1 Select the step before which you want the new (second) action to begin.

2 Choose **Step** > **Split Action** or click the Split Action button. The Split Action dialog box opens:



Note: The Split Action option is disabled in the following situations:

- when an external action is selected
- when the first line of the action is selected
- while recording a test
- while running a test
- when you are working with a read-only test

---

3. Choose **Independent of each other** or **Nested (the second Action is called by the first)** to determine whether the new action will become a top-level (sibling) action or a nested (child) action.
4. If you wish, modify the name and description of the first action.
5. If you wish, modify the name and description of the second action.

Note: If a reusable action is called more than once in a test, and you split the action into two independent actions, each call to the action within the test will be followed by a call to the new (reusable) action. If a reusable action is called from another test, however, splitting it may cause the calling test to fail.

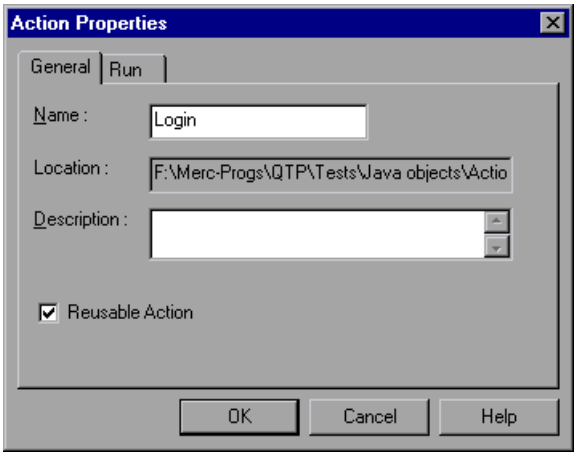
Setting Action Properties

The Action Properties dialog box enables you to modify an action name, add a description for an action, set an action as a reusable action, set the run properties for an action, and set the parameter properties (for external actions only).

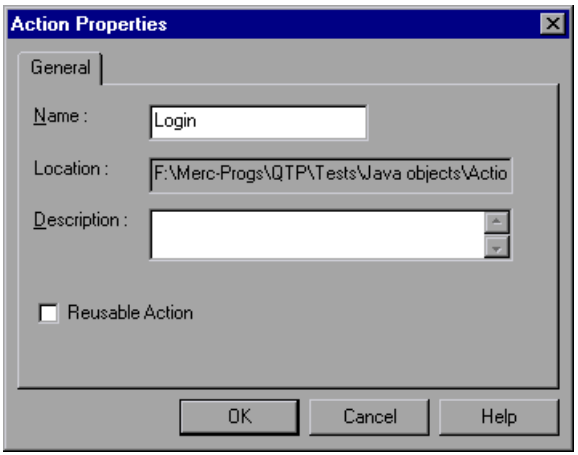
Opening the Action Properties Dialog Box

You can open the Action Properties dialog box from the Tree View with the test flow displayed, from the Tree View with an action view displayed or from the Expert View (displays the script of the current action).

When you open the Action Properties dialog box in the Tree View with the test flow displayed (either from a test without reusable or external actions, or with **Test Flow** selected in the Action List), the Action Properties dialog box contains the General tab and the Run tab as shown below:



When you open the Action Properties dialog box from an action view (either from the Tree View or the Expert View with a specific action displayed in the Action List), the Action Properties dialog box contains only the General tab as shown below:



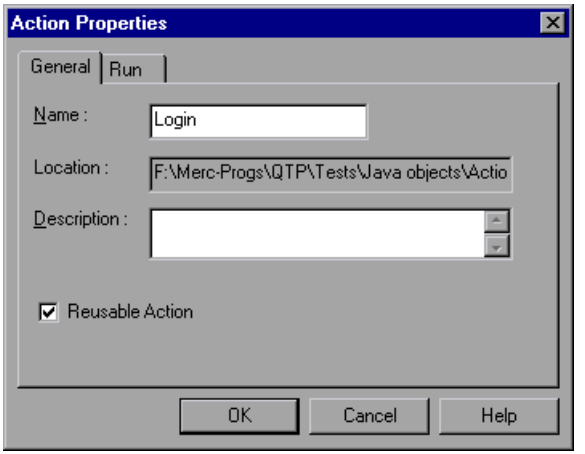
Note: In addition to the tabs described above, the Action Properties dialog box for an external action contains a Parameters tab. For more information, see "Setting the Parameter Storage Properties For an External Action," on page 256.

Adding or Editing an Action Description

When you add a description to an action, the description is displayed in the action's Action Properties dialog box. An action description helps you and other testers know what a specific action does without reviewing the entire script or expanded tree of the action. The description is also displayed in the description section of the Insert Copy of Action and Insert Call to Action dialog boxes. This enables you and other testers to determine which action you want to insert from another test without having to open it. For more information about inserting copies of actions and calls to actions, see "Inserting Existing Actions," on page 240.

To add or edit an action description:

1 Right-click the external action in the test tree or anywhere in the Expert View and select **Action Properties,** or choose **Step** > **Action Properties**. The Action Properties dialog box opens.



2 Enter or modify the description of the action in the **Description** box.

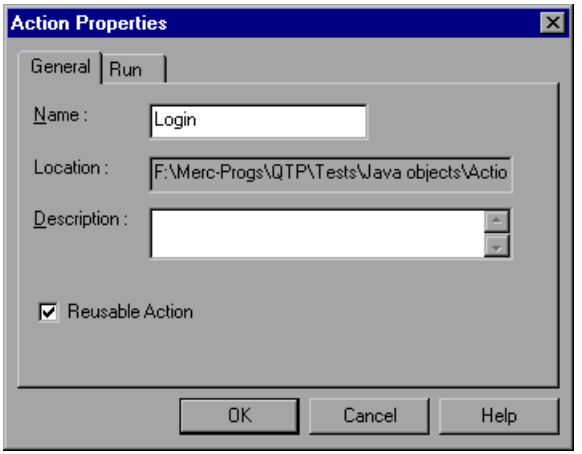
Note: You can also add a description when inserting a new action. For more information about adding a new action, see "Creating New Actions," on page 238.

Setting the Reusability Status of an Action

A reusable action can be called multiple times within a test, and can be called from other tests. Non-reusable actions can be copied and inserted as independent actions, but cannot be inserted as calls to the original action.

To change an action's reusability status:

1 Right-click the external action in the test tree or anywhere in the Expert View and select **Action Properties,** or choose **Step > Action Properties**. The Action Properties dialog box opens.



2 Select or clear the **Reusable Action** check box.

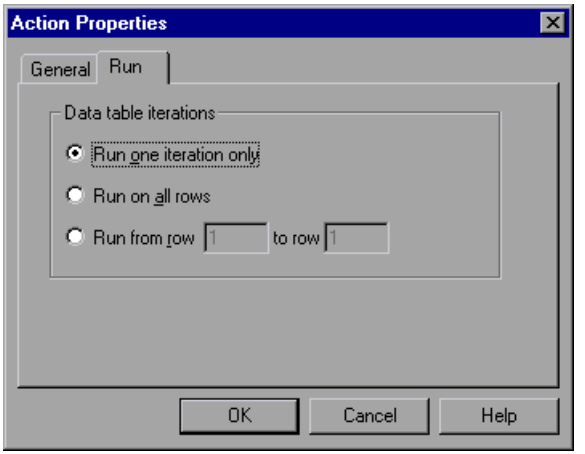
Note: If the test contains more than one call to the action, the **Reusable Action** option cannot be cleared. If you want to make the action non-reusable, remove the additional calls to the action.

3 Click **OK**. If the action tree was expanded, it collapses, and the action icon changes to a reusable or non-reusable action icon, as appropriate.

Note: You cannot expand reusable actions from the test flow view. You can view details of a reusable action in the Tree View by switching to the action view for that action. For more information about the test flow and action views, see "Using the Action Toolbar," on page 237.

Setting the Run Properties for an Action

You can use the Action Properties Run tab to instruct QuickTest to run only one iteration on the action, to run iterations on all rows in the data table, or to run iterations only for certain rows in the data table.



The Run tab includes the following options:

Option	Description
Run one iteration only	Runs the action only once, using the row that corresponds to the global iteration counter. For example, suppose an action's data sheet has two rows and the global sheet has four rows. If you choose to run one iteration only for the action and you choose to run iterations on all rows of the global data sheet, then during each iteration of the test, this action will run only one iteration. The data that the action parameters use during each repetition of the test are based on the iteration number for the test. During the first iteration of the test, action parameters in the action will take data from the first row of the action's data sheet. In the second iteration of the test, action parameters in the action will take data from the second row of the action's data sheet. In the third and subsequent iterations of the test, the action's parameters in the action will continue to take data from the second (last) row of the action's data sheet.
Run on all rows	Runs the action with iterations using all rows in the action's data table.
Run from row __ to row __	Runs the action with iterations using the values in the action's data table for the specified row range.

Notes: If you run multiple iterations on an action, the action must begin and end on the same Web page or frame, so that it is in the proper location to run the next iteration of the action.

The Run tab of the Action Properties dialog box applies to individual actions and refers to the rows in the action's data sheet. You can set the Run properties for an entire test (setting iterations for rows on the global data sheet) from the Run tab in the Test Settings dialog box. For more information, see Chapter 25, "Setting Testing Options for a Single Test."

Setting the Parameter Storage Properties For an External Action

When you insert an external action, you choose whether you want to use the parameterization data from the original action, or whether you want to insert an editable copy of the parameterization data into the test's action data table. You can use the Action Properties Parameters tab to change the parameter storage setting.

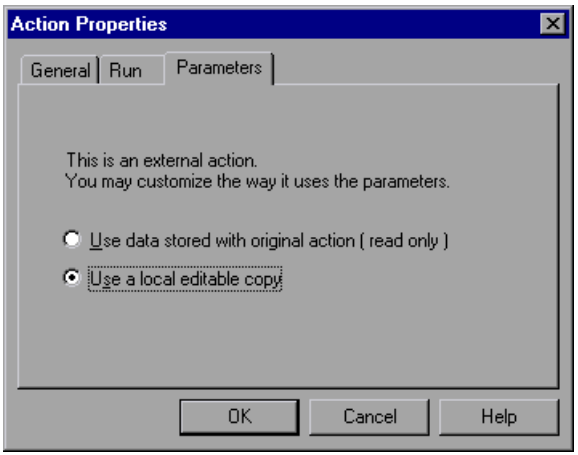
To modify the parameter storage settings:

1 Right-click the external action in the test tree and select **Action Properties,** or choose **Step** > **Action Properties** from the Tree View or Expert View. The Action Properties dialog box opens.

Note: If you open the Action Properties dialog box from the Expert View, the dialog box does not include the Run tab.

---

2 Click the Parameters tab.



3 Choose where you want to store the parameterization data:

- To use the original action's data, select **Use data stored with the original action (read-only)**. When you select this option, the data is read-only when viewed from the calling test and all changes to the original action's data sheet apply when the action runs in the calling test.
- To store an editable copy of the data in the test's data table, select **Use a local, editable copy**. When you select this option, the data sheet is stored in the test's data table and is independent of the original action. Changes to the original data sheet do not affect the calling test.

Exiting an Action

You can add a line in your script in order to exit an action before it runs in its entirety. You may want to use this option in order to return the current value of the action at a specific point in the run. There are four types of exit action statements you can use:

- **ExitAction** - Exits the current action, regardless of its iteration attributes.
- **ExitActionIteration** - Exits the current iteration of the action.
- **ExitRun** - Exits the test, regardless of its iteration attributes.
- **ExitGlobalIteration** - Exits the current global iteration.

You can view the exit action node in the Test Results tree, and if your exit action statement returns a value, the value is displayed in the action, iteration, or test summary, as applicable.

For more information about these functions, refer to the *QuickTest Object Model Reference*.

For more information about the Test Results, see Chapter 18, "Analyzing Test Results."

Removing Actions From a Test

The procedures and effects of removing non-reusable actions, external actions, reusable actions, or calls to external or reusable actions are different.

- When you remove a non-reusable action, you delete the action entirely.
- When you remove a call to a reusable or external action, you remove the action from your test flow, but the action remains part of the test.
- When you remove an external action, you remove the action from your test entirely. The original action is not affected.
- When you remove a reusable action, you delete the action entirely. This will cause any test calling this action to fail.

Removing a Non-Reusable Action

Removing a non-reusable action from your test deletes the action entirely.

To remove a non-reusable action:

1. Select the action you want to remove and press the **Delete** key on your keyboard or select **Edit > Delete**. A delete confirmation message box opens.
2. Click **Yes** to confirm.

Removing a Call to a Reusable or External Action From the Test Flow

You should choose to remove a call to a reusable or external action if you want to remove the action from the test flow, but you still want the action to be available for other calls. When you choose this option, the action still exists even though it is removed from your test flow, and the action's data table tab remains. You can still view the action (and edit a reusable action) by selecting it from the Action List in the Tree View or in the Expert View.

After you remove a call to an action, you can insert the action back into this test or into any other test using the **Insert Copy of Action** or **Insert Call to Action** options. For more information see "Inserting Existing Actions," on page 240.

To remove a call to a reusable or external action from the test flow:

1. Select the **Test Flow** view from the Action List in the Tree View.
2. Highlight the action you want to remove.
3. Choose **Edit** > **Delete**, or press the **Delete** key on your keyboard. The Confirm Action Removal message box opens.
4. Click **Yes** to close the message box and remove the call to the action.

Removing an External Action from the Test

When you remove an external action from the test, the action is removed in its entirety, including the corresponding action sheet in the data table. Columns related to this action that are located in the global sheet are not removed.

After you remove an external action from your test, you can reinsert it by choosing **Insert** > **Call to Action** and locating the test where the original action resides. For more information see “Inserting Existing Actions,” on page 240.

To remove an external action from the test:

1. Select the action you want to remove from the Action List.
2. Highlight the action icon in the test tree.
3. Choose **Edit** > **Delete**, or press **Delete** on your keyboard. The Confirm Action Removal message box opens.
4. Click **Yes** to close the message box and delete the action.

Removing a Reusable Action from the Test

You should choose to remove a reusable action from the test only if you are sure that you no longer need the action, and that no other test calls this action. This option deletes the action entirely.

Note: Deleting a reusable action that has been called by other tests will cause those tests to fail.

To remove a reusable action from the test:

1. Select the action you want to remove from the Action List.
2. Highlight the action icon in the test tree.
3. Choose **Edit** > **Delete** or press **Delete** on your keyboard. The Confirm Action Removal message box opens.
4. Click **Yes** to close the message box and delete the action.

Guidelines for Working with Actions

Consider the following guidelines when working with actions:

- When you parameterize an action in your test, the action must 'clean up after itself.' In other words, the action must end at the same point it started, so that it can run again without interruption. For example, suppose you are testing the sample flight reservation site. If the action starts with a blank flight reservation form, it should conclude with a blank flight reservation form.
- A single test may include both global parameterization and action parameterization. For example, you can create a test in which a travel agent logs into the flight reservation system, books three flights, and logs out; the next travel agent logs into the flight reservation system, books three flights, and logs out, and so on. To parameterize the 'book a flight' action, you choose **Current action sheet (local)** in the parameterization dialog box and enter the three flights into the relevant **Action** tab in the Data pane. To parameterize the entire test, you choose **Global** in the parameterization dialog box and enter the login names and passwords for the different agents into the **Global** tab in the Data pane.
- Your entire test will run one time for each row in the global data sheet. Within each test, each parameterized action will be repeated according to the number of rows in its data sheet.
- You may want to rename the actions in your test with descriptive names to help you identify them. It is also a good idea to add detailed action descriptions. This facilitates inserting actions from one test to another. You can rename an action by choosing **Edit** > **Rename Action**.

- If you plan to use an identical or virtually identical procedure in more than one test, you should consider inserting an action from another test. If you want to make slight modifications to the action in only one test, you should use the **Insert Copy of Action** option to paste a copy of the action. If you want modifications to affect all tests containing the action, you should use the **Insert Call to Action** option to insert a link to the action in the original test.
- Reusable actions help you to maintain your tests, but it is important to consider the effects of making an action reusable. Once you make an action reusable, be sure to consider how changes to your action could potentially affect other tests that call that action.
- If you expect certain elements of your Web site to change regularly, it is a good idea to divide the steps related to changeable elements into a separate action so that it will be easy to re-record the required steps after the site is modified.
- Use the global data table to pass parameters from one action to another. For information on the global data table, see Chapter 15, "Working with Data Tables."

15

Working with Data Tables

QuickTest enables you to create and run tests that are driven by data stored in the data table.

This chapter describes:

- Global and Action Sheets
- Editing the Data Table
- Importing Data from a Database
- Using Formulas in the Data Table
- Using Data Table Scripting Methods

About Working with Data Tables

You can insert data table parameters and output values into your test so that it will run several times on different sets of data. Each test run on a different set of data is called an *iteration*. The data your test uses is stored in the data table, which is displayed in the Data pane at the bottom of the screen. The data table has the characteristics of a Microsoft Excel spreadsheet, meaning that you can also execute mathematical formulas within the cells.

After you run a test, the data you enter in the data table is displayed in the Runtime data table within the Test Results window. For more information on the Runtime data table, see “Viewing the Runtime Data Table,” on page 311.

Global and Action Sheets

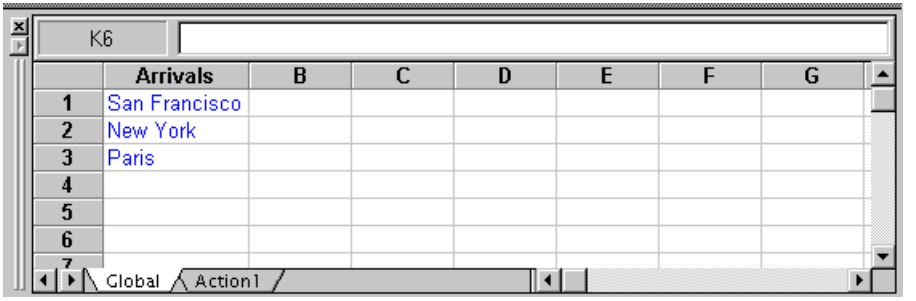
There are two types of sheets within the data table: *Global* and *Action*. You can access the different sheets by clicking the appropriate tabs below the data table.

- You store data in the Global tab when you want it to be available to all actions in your test, for example, to pass parameters from one action to another.
- You store data in the action's tab when you want it to be available to only one action in your test.

For example, suppose you are creating a test on the sample Flight Reservation Web site. You might create one action for logging in, another for booking flights, and a third for logging out. You may want to create a test in which the user logs onto the site once, and then books flights for five passengers. The data about the passengers is relevant only to the second action, so it should be stored in the action tab corresponding to that action.

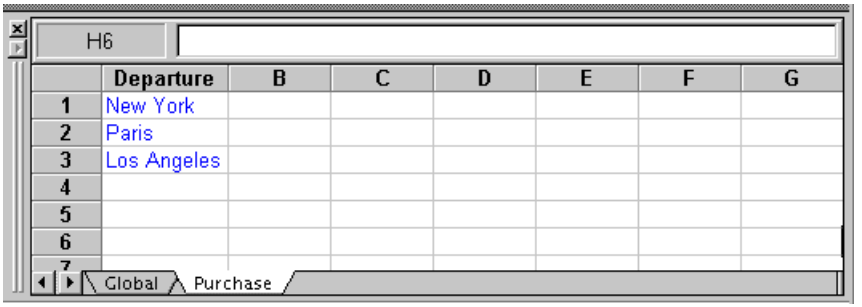
Global Sheet

The Global sheet contains the data that replaces parameters in each iteration of the test. If you create a Global parameter called Arrivals, the Global sheet might look like this:



Current Action Sheets

Each time you add a new action to the test, a new *Action* sheet is added to the data table. Action sheets are automatically labeled with the exact name of the corresponding action. The data contained in an action sheet is relevant for the corresponding action only. For example, if a test had the data table below, QuickTest would use the data contained in the Purchase sheet when running iterations on steps with action parameters within the *Purchase* action:



For more information about creating global and action parameters, see Chapter 10, “Parameterizing Tests.”

Editing the Data Table

The data table contains the values that QuickTest substitutes for parameters when you run a test. Whenever you save your test, QuickTest automatically saves the test’s data table as an *.xls* file.

By default, the data table is saved in the test folder. You can save the data table in another location (and instruct the test to use this data table when running the test) by specifying it in the Properties tab of the Test Settings dialog box. This can be useful if you want to run the same test with different sets of input values. For example, you can test the localization capabilities of your application by running your test with a different data table file for each language you want to test. For more information on the Test Settings dialog box, see Chapter 25, “Setting Testing Options for a Single Test.”

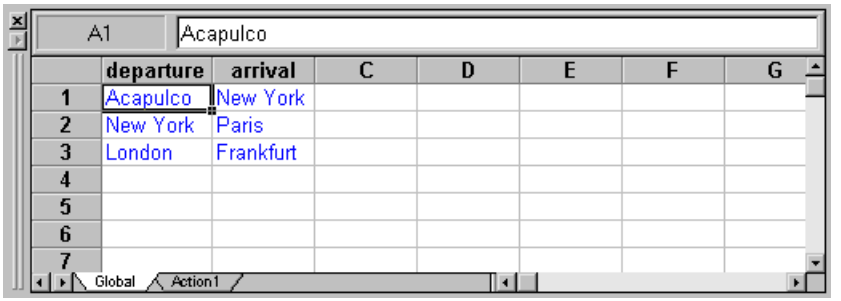
Tip: You can also vary the user interface strings that you check in each language by using a different environment parameter file each time you run the test. For more information, see Chapter 10, "Parameterizing Tests."

You can edit information in the table by typing directly into the table. You can also import data in Excel 95, Excel 97, Excel 2000, or ASCII format. You use the table in the same way as an Excel spreadsheet, including inserting formulas into cells.

To edit the data table:

1 Open your test.

2 Make sure the **Data Views** button is enabled.



- Each *row* in the table represents the set of values that QuickTest submits for the parameterized arguments during a single iteration of the test or action. The number of iterations that a test runs is equal to the number of rows in the table.
- Each *column* in the table represents the list of values for a single parameterized argument. The column header is the parameter name.

Note: You must enter data in rows from top to bottom, i.e., you cannot enter data in a cell in a row until you have entered data in a previous row.

3 To change the name of a column, double-click the column heading cell. The Change Parameter dialog box opens. Type a parameter name and click **OK**.

Note: If you change the name in the table, you must also change the corresponding parameter name in the test pane.

---

4 Use the data table menu commands described below to edit the table. To open the data table menu, right-click a cell. The following menus are available: File, Edit, Data, and Format.

File Menu

The following commands are available in the File menu:

File Command	Description
Import From File	Imports an existing Excel table file into the table. **Note:** The table file you import replaces all data in all sheets of the table, and the first row in each Excel sheet replaces the column headers in the corresponding data table sheet. It is therefore essential that the first row of your data table exactly matches the parameter names in your test.
Import From Database	Imports data from the specified database.
Export	Exports the table to a specified excel file.
Print	Prints the table.

Edit Menu

The following commands are available in the Edit menu:

Edit Command	Description
Cut	Cuts the table selection and puts it on the Clipboard.
Copy	Copies the table selection and puts it on the Clipboard.

Edit Command	Description
Paste	Pastes the contents of the Clipboard to the current table selection.
Paste Values	Pastes values from the Clipboard to the current table selection. Any formatting applied to the values is ignored. In addition, only formula results are pasted; formulas are ignored.
Clear	Clears formats or contents from the current selection. You can clear only formats, only contents (including formulas), or both formats and contents.
Insert	Inserts empty cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells.
Delete	Deletes the current selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells.
Fill Right	Copies data in the left-most cell of the selected range to all cells to the right of it within the selected range.
Fill Down	Copies data in the top cell of the selected range to all cells below it within the selected range.
Find	Finds a cell containing specified text. You can search by row or column in the table and specify to match case or find entire cells only.
Replace	Finds a cell containing specified text and replaces it with different text. You can search by row or column in the table and specify to match case and/or to find entire cells only. You can also replace all.
Go To	Goes to a specified cell. This cell becomes the active cell.

Data Menu

The following commands are available in the Data menu:

Data Command	Description
Recalc	Recalculates any formula cells in the table.
Sort	Sorts a selection of cells by row and/or column and keys.
AutoFill List	Creates, edits, or deletes an autofill list. An autofill list contains frequently-used series of text such as months and days of the week. When adding a new list, separate each item with a semi-colon. To use an autofill list, enter the first item into a cell in the table. Drag the cursor, from the bottom right corner of the cell, and QuickTest automatically fills in the cells in the range according to the autofill list.

Format Menu

The following commands are available in the Format menu:

Format Command	Description
General	Sets format to General. General displays numbers with as many decimal places as necessary and no commas.
Currency(0)	Sets format to currency with commas and no decimal places.
Currency(2)	Sets format to currency with commas and two decimal places.
Fixed	Sets format to fixed precision with commas and no decimal places.
Percent	Sets format to percent with no decimal places. Numbers are displayed as percentages with a trailing percent sign (%).
Fraction	Sets format to fraction.
Scientific	Sets format to scientific notation with two decimal places.
Date: (dynamic)	Sets format to Date with the M/d/yy format.
Time: h:mm AM/PM	Sets format to Time with the h:mm AM/PM format.

Format Command	Description
Custom Number	Sets format to a custom number format that you specify.
Validation Rule	Sets validation rule to test data entered into a cell or range of cells. A validation rule consists of a formula to test, and text to display if the validation fails.

Importing Data from a Database

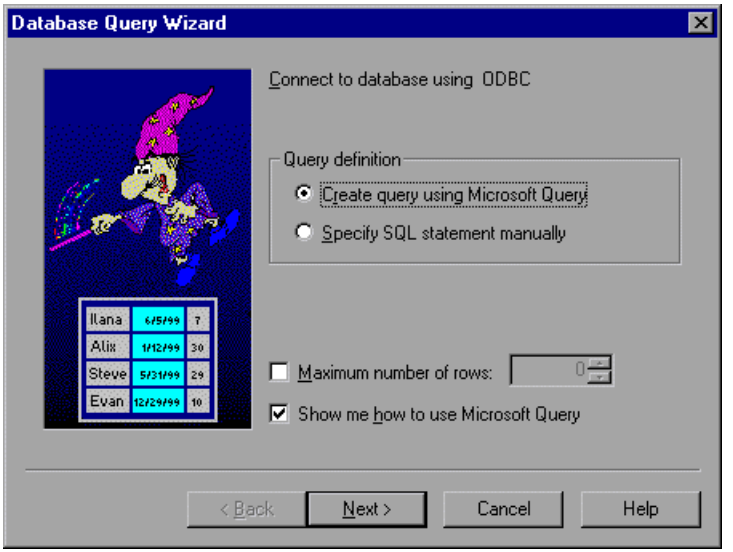
You can import data from a database by selecting a Query from Microsoft Query or by manually specifying an SQL statement.

You can install Microsoft Query from the *custom installation* of Microsoft Office.

Note: Contrary to importing an excel file (**File > Import > From File**), existing data in the data table is *not* replaced when you import data from a database. If the database you import contains a column with the same name as an existing column, the database column is added as a new column with the column name followed by a number. For example, if your data table already contains a column called departures, a database column by the same name would be inserted into the data table as: departures1.

To import data from a database:

1 Right-click on the data table sheet to which you want to import the data and select **File** > **Import** > **From Database**. The Database Query Wizard opens.

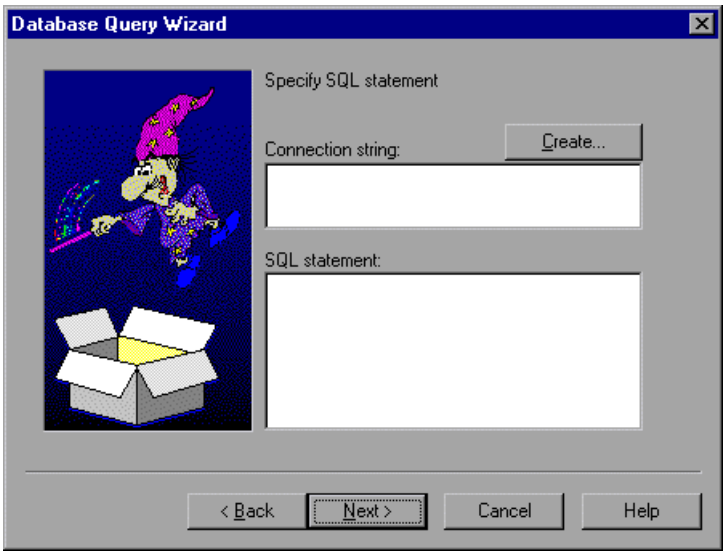


2 Select your database selection preferences and click **Next**. You can choose from the following options:

- **Create query using Microsoft Query:** Opens Microsoft Query, enabling you to create a new query. Once you finish defining your query, you are exit back to QuickTest.
- **Specify SQL statement manually:** Opens the **Specify SQL statement** screen in the wizard, which enables you to specify the connection string and an SQL statement. For additional information, see step 3.
- **Maximum number of rows:** Select this check box and enter the maximum number of database rows to check. You can specify a maximum of 32,000 rows.
- **Show me how to use Microsoft Query:** Displays an instruction screen before opening Microsoft Query, when you click **Next**. (Enabled only when **Create query using Microsoft Query** is selected).

3 If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information about creating a query, see "Creating a Query in Microsoft Query," on page 273.

If you chose **Specify SQL statement manually** in the previous step, the following screen opens:



Specify the connection string and the SQL statement, and click **Next**.

- **Connection string:** Enter the connection string, or click **Create** to open the ODBC Select Data Source dialog box. You can select a *****.*dsn* file in the ODBC Select Data Source dialog box to have it insert the connection string in the box for you.
- **SQL statement:** Enter the SQL statement.

4 QuickTest takes several seconds to capture the database query and restore the QuickTest window. The data from the database is displayed in the data table.

Creating a Query in Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source. QuickTest supports the following versions of Microsoft Query:

- version 2.00 (in Microsoft Office 95)
- version 8.00 (in Microsoft Office 97)
- version 2000 (in Microsoft Office 2000)

To choose a data source and define a query in Microsoft Query:

1 When Microsoft Query opens during the Import data from database process, choose a new or an existing data source.

2 Define a query.

3 When you are done:

- Version 2.00: Choose **File** > **Exit and return to QuickTest** to close Microsoft Query and return to QuickTest.
- Version 8.00 and Version 2000: In the Finish screen of the Query Wizard, select **Exit and return to QuickTest** and click **Finish** to exit Microsoft Query. Alternatively, click **View data or edit query in Microsoft Query** and click **Finish**. After viewing or editing the data, choose **File** > **Exit and return to QuickTest** to close Microsoft Query and return to QuickTest.

4 Return to step 4 in "Importing Data from a Database," on page 270 to continue creating your database checkpoint in QuickTest.

For additional information on working with Microsoft Query, refer to the Microsoft Query documentation.

Using Formulas in the Data Table

You can use any Microsoft Excel formula in your data table. This enables you to create contextually relevant data during the test run. You can also use formulas as part of a checkpoint to check that objects from a page created on-the-fly (dynamically generated) or other variable objects in your Web page have the values you expect for a given context.

When you use formulas in a data table to compare values (generally in a checkpoint), the values you compare must be of the same type, i.e. integers, strings, etc. When you extract values from different places in your applications using different functions, the values may not be of the same type. Although these values may look identical on the screen, a comparison of them will fail, since, for example, 8.2 is not equal to "8.2". You can use the **TEXT** and **VALUE** functions to convert values from one type to another:

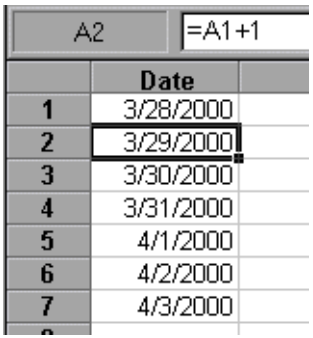
- **TEXT(value)** returns the textual equivalent of a numeric value, so that, for example, TEXT(8.2)="8.2";
- **VALUE(string)** returns the numeric value of a string, so that, for example, VALUE("8.2")=8.2.

For additional information on using worksheet functions, refer to the *Microsoft Excel* documentation.

Using Formulas to Create Parameterization Data

You can enter formulas rather than fixed values in the cells of a parameter column.

For example, suppose you want to parameterize the value for a WebEdit object that requires a date value no earlier than today's date. You can set the cells in the Date column to the date format, and enter the =NOW() Excel formula into the first row in order to set the value to today's date for the first iteration. Then you can use another formula in the rest of the rows in order to enter the above date plus one day, as shown below. By using this formula you can run the test on any day, and the dates will always be valid.

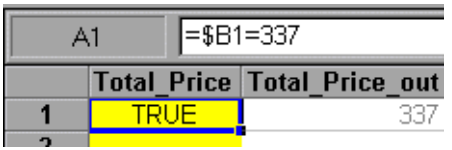


Using Formulas in Checkpoints

You can use a formula in a checkpoint in order to confirm that an object from a page created on-the-fly (dynamically generated) or another variable object in your Web page contains the value it should for a given context. For example, suppose a shopping cart Web site displays a price total. You can create a text checkpoint on the displayed total value and use a data table formula to check whether the site properly computes the total, based on the individual prices of the products selected for purchase in each iteration.

When you use the data table formula option with a checkpoint, QuickTest creates two columns in the data table. The first column contains a default checkpoint formula. The second column contains the value to be checked in

the form of an output parameter. The result of the formula is Boolean: TRUE or FALSE.



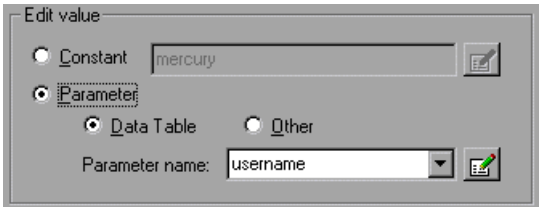
A FALSE result in the checkpoint column during a test run causes the test to fail.

Once you finish adding the checkpoint, you can modify the default formula in the first column to perform the check you need.

To use a formula in a checkpoint:

1 Select the page, text, or object for which you want to create a checkpoint and open the Insert Checkpoint dialog box as described in Chapter 6, "Creating Checkpoints."

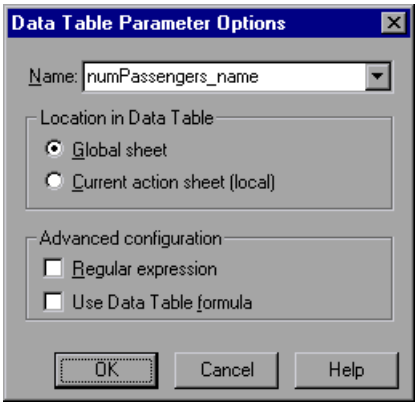
2 In the Edit value box, click **Parameter**.



3 Click **Data Table** and choose a parameter from the **Parameter name** box list or enter a new name.

- To use an existing parameter, select it from the list.
- To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter.

4 Click the **Edit Parameter Options** button. The Data Table Parameter Options dialog box opens.



5 Select the **Use Data Table formula** check box and click **OK** to close the Data Table Parameter Options dialog box.

6 Specify your other checkpoint setting preferences as described in Chapter 6, "Creating Checkpoints."

7 Click **OK.** The two columns are added to the table, and a checkpoint icon is added to your test tree.

8 Highlight the value in the first (formula) column to view the formula and modify the formula to fit your needs.

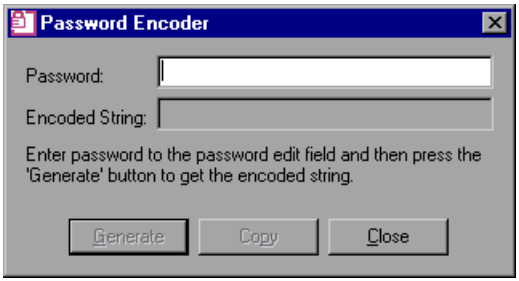
9 If you want to run several iterations, add the appropriate formula in subsequent rows of the formula column for each iteration in the test or action.

Inserting Encoded Passwords into the Data Table

You can encode passwords in order to use the resulting strings as parameter values. 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 and place secure values into the data table.

To encode a password:

1 Select **Start** > **Programs** > **QuickTest** > **Tools** > **Password Encoder**. The Password Encoder dialog box opens.



2 Enter the password in the **Password** box.

3 Click **Generate**. The Password Encoder encrypts the password and displays it in the **Encoded String** box.

4 Use the **Copy** button to copy and paste the encoded value in the data table.

5 Repeat the process for each password you want to encode.

6 Click **Close** to close the Password Encoder.

Using Data Table Scripting Methods

QuickTest provides several data table methods that enable you to retrieve information about the data table and to set the value of cells in the data table.

You enter these statements manually in the Expert View. For more information about working in the Expert View see Chapter 22, "Testing in the Expert View."

From a programming perspective, the data table is made up of three types of objects: DataTable, Sheet (sheet), and Parameter (column). Each object has several methods and properties that you can use to retrieve or set values.

For more details on the data table methods, refer to the *QuickTest Object Model Reference*.

16

Handling Unexpected Events and Errors

You can instruct QuickTest to handle unexpected events and errors that occur in your testing environment during a test run.

This chapter describes:

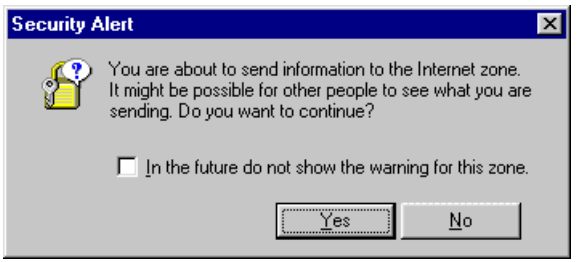
- Changing the Status of Exceptions
- Modifying Exceptions
- Adding New Exceptions
- Deleting Exceptions
- Configuring Event Handling

About Handling Unexpected Events and Errors

Unexpected events and errors during a test run can disrupt your test and distort test results. This is a problem particularly when running tests unattended: the tests are suspended until you perform the action needed to recover.

You can use the *Exception Editor* to instruct QuickTest to detect and handle the appearance of a specific dialog box and act to recover the test run.

For example, if a Security Alert dialog box is displayed during a test run that does not have a corresponding step in your test, you can instruct QuickTest to recover the test run by clicking the default button. In this particular case, the Yes button is the default button.



The Exception Editor contains a list of exceptions that QuickTest supports. Each exception is associated with the action that is activated in order to recover the test run. You can modify the list of exceptions and configure additional types of dialog box exceptions that you would like QuickTest to support.

Note: Exception handling helps in situations where events occur during a test run that do not have a corresponding step in your test. If you have a step in your test related to a dialog box that may not always display, you can choose to bypass the step containing the dialog box using an optional step. For additional information, see Chapter 17, "Running Tests."

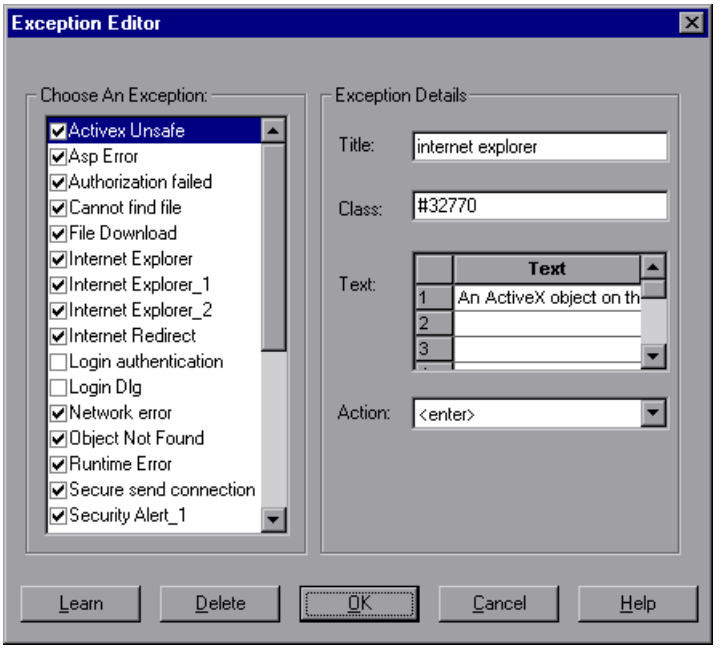
QuickTest stores information about handling unexpected events and errors in an external exception configuration file. You can use this file to "learn" exceptions on one machine and "teach" them to other machines. You can also use this file to handle exceptions when testing localized versions of a single application.

Changing the Status of Exceptions

The Exception Editor includes a list of all the available exceptions. You can choose to activate or deactivate any exception in the list.

To change the status of an exception:

1 Choose **Tools** > **Exception Editor**. The Exception Editor opens.



2 In the **Choose An Exception** list, click an exception.

The exception is highlighted. The current description of the exception is displayed in the Exception Details area.

3 To activate an exception, select its check box. To deactivate the exception, clear its check box.

4 Click **OK** to save the changes.

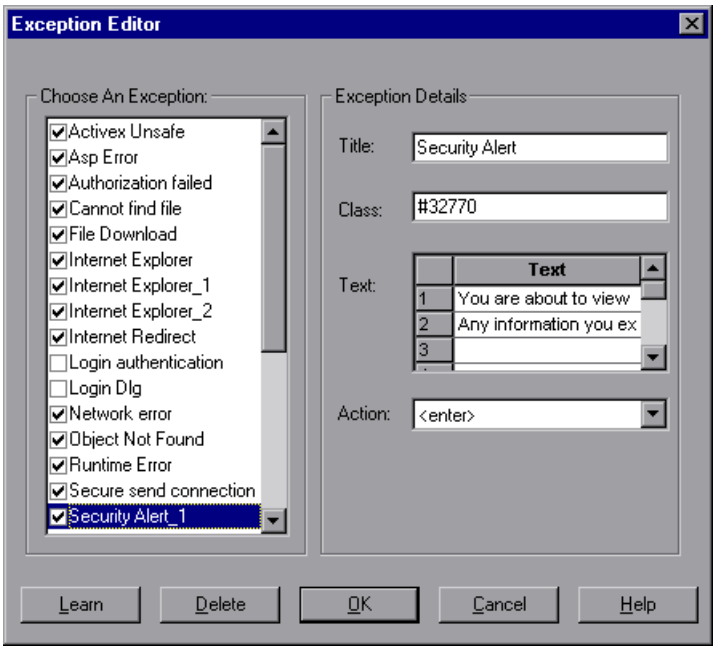
Modifying Exceptions

You can modify the details of an exception listed in the Exception Editor.

To modify the details of an exception:

1 Choose **Tools** > **Exception Editor**. The Exception Editor opens.

2 In the **Choose An Exception** list, click an exception.



The exception is highlighted. The current description of the exception is displayed in the Exception Details area.

3 You can modify the title or the content of the exception dialog box, or which handler function to associate with it.

- To modify the title of the exception dialog box, edit the text in the **Title** box.
- To modify the text that is displayed in the exception dialog box, edit a text line in the **Text** box.
- To change the action associated with this exception dialog, choose a function from the **Action** list. This is the function that recovers the test run.

Action	Description
<enter>	Presses the **Enter** key.
<login>	Uses the user name and password you supply in the **User Name** and **Password** edit boxes, which are displayed when you select this function.
<press button>	Clicks the button you select from the **Button Name** list, which is displayed when you select this function.

4 Click **OK** to save the changes.

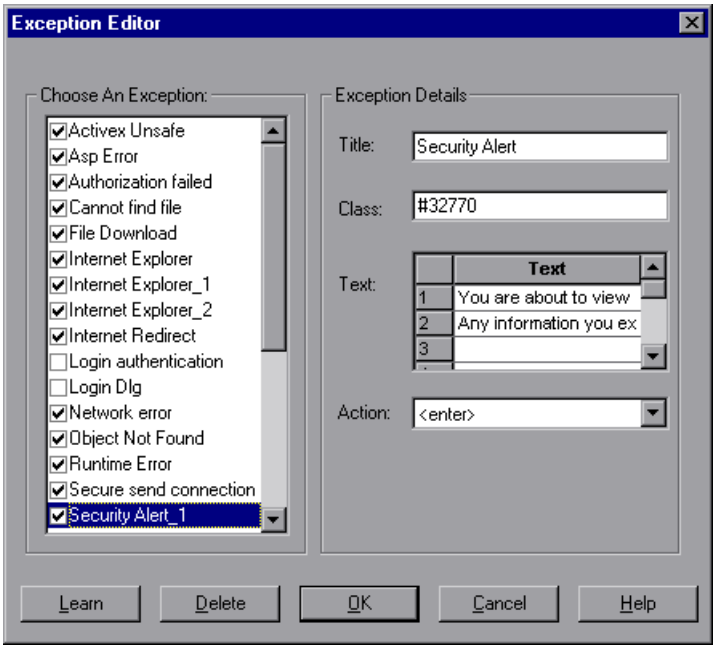
Adding New Exceptions

You can add a new exception to the list of exceptions in the Exception Editor.

To add a new exception:

1 Choose **Tools** > **Exception Editor**. The Exception Editor opens.

2 Click the **Learn** button. The mouse pointer becomes a pointing hand. Click the dialog box for which you want to add an exception. A new exception is added to the list.



The Exception Editor displays the title of the exception dialog box, the class of the exception, the text that is displayed in the dialog box, and action that is responsible for recovering test execution. To modify these fields, refer to "Modifying Exceptions" on page 284.

3 Click **OK** to save the exception.

Deleting Exceptions

You can delete an exception from the list of exceptions in the Exception Editor.

To delete an exception:

1. Choose **Tools** > **Exception Editor**. The Exception Editor opens.
2. In the **Choose An Exception list**, click an exception to delete.

   The exception is highlighted. The current description of the exception is displayed in the Exception Details area.
3. Click the **Delete** button. A warning dialog box opens.
4. Click **Yes** to delete the exception.
5. Click **OK** to exit the Exception Editor.

Configuring Event Handling

QuickTest stores information about handling unexpected events and errors in an external exception configuration file. This file, *Exception.inf*, is located in the *[QuickTest installation]\dat* folder.

You may want to access this file for the following reasons:

➤ to "learn" exceptions on one machine and "teach" them to other machines

---

Tip: Set the exception handling as described in this chapter on one machine. Then replace the file on the other machine(s) with the new exception file.

➤ to handle exceptions when testing localized versions of a single application

---

Tip: Set the exception handling as described in this chapter for each localized version. When you change the localized version of the application you are testing, you simply switch the *Exception.inf* exception configuration file.

Part IV

Running and Debugging Tests

17

Running Tests

Once you have created a test, you run it to check the behavior of your Web site.

This chapter describes:

- Running a Test to Check Your Application
- Running a Test to Debug It
- Updating a Test
- Using Optional Steps
- Running a Test Batch

About Running Tests

When you run a test, QuickTest navigates through the Web-based application performing the steps you recorded. Once the test run is complete, QuickTest displays a report detailing the test results.

If your test contains a global data table parameter, QuickTest runs the test once for each row of data in the Data pane. If your test contains a data table parameter in the current action tab, QuickTest runs the action once for each row of data in the Data pane. You can also specify whether to run the first iteration or all iterations for the test, or a specific action, or to run the iterations for a specified range of data sets. For additional information, see Chapter 25, "Setting Testing Options for a Single Test" and Chapter 14, "Working with Actions."

You can set up a batch of tests and run them sequentially using the QuickTest Test Batch Runner. For more information, see "Running a Test Batch," on page 298.

You can also run tests on objects with dynamic descriptions. For additional information, see Chapter 4, "Managing Test Objects."

Note for WinRunner users: You can run WinRunner tests and call functions from WinRunner compiled modules while running an QuickTest test. For additional information, see Chapter 29, "Working with WinRunner."

Running a Test to Check Your Application

When you run a test, QuickTest performs the steps you recorded on your Web site and displays each Web page in your browser. QuickTest always runs a test from the first step in the test.

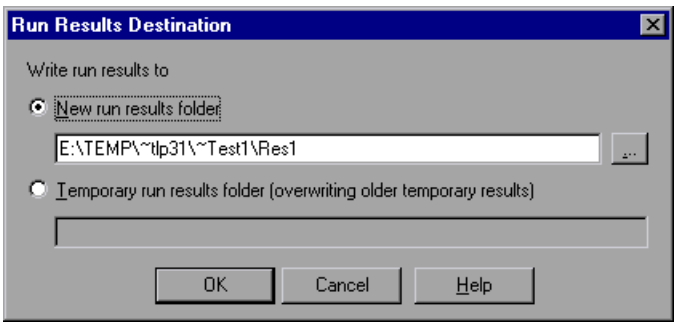
If your test does not contain parameterized values, QuickTest runs the test once. If the test does contain parameters, QuickTest runs the test for each row in the table in the Data pane, using the parameters you specified. If an action within your test is parameterized, you can choose to set and run only certain data sets. For more information, see Chapter 14, "Working with Actions."

Note: If you are running a test in a TestDirector project from QuickTest and you want to save the test results to the project, see "Saving Test Results to a Project," on page 454.

To run a test to check your application:

1 If your test is not already open, choose **File** > **Open** or click the **Open** button to open the test.

2 Click the **Run** button on the toolbar, or choose **Test** > **Run**. The Run Results Destination dialog box opens with the New run results folder option selected by default. This option displays the default path and a folder name for the test run results.



3 To save the test run results under a different name, type it in the text box or click the browse button to locate the folder.

To run the test and overwrite the previous test run results, click **Temporary run results folder**.

Note: QuickTest stores temporary test run results for all tests in <*System Drive:**Temp**TempResults*>. The path in the text box of the **Temporary run results folder** option is read-only and cannot be changed.

---

4 Click **OK**. The Run Results Destination dialog box closes and QuickTest begins running the test. As QuickTest runs the test, it highlights each step in the test tree.

When the test stops running, the Test Results window opens unless you have cleared the **View results when test run ends** check box in the Run tab of the Options dialog box. For more information about the Options dialog box, see Chapter 24, “Setting Global Testing Options.”

Note: If you want to interrupt a test that is running, you can:

Click the **Pause** button or choose **Debug** > **Pause**. The test run pauses. To resume running a paused test run, click the **Run** button or choose **Test** > **Run**.

Click the **Stop** button or choose **Test** > **Stop**. The test run stops running and the **Test Results** window opens.

Running a Test to Debug It

When debugging a test, you can run it from the selected step until the end of the action in which it is contained.

If the action contains nested actions, QuickTest runs them for the defined number of iterations. For information on nested actions, see Chapter 14, "Working with Actions."

At the end of the test run, a report will be produced if you select the **View results when test run ends** check box in the Run tab of the **Tools** > **Options** menu.

To run a test to debug it:

1 Mark the place where you want to start running the test:

- In the Tree View, highlight a step.
- In the Expert View, place your cursor in a line of VBScript.

2 Choose **Test** > **Run Action From Step**.

3 In the Run Results Destination dialog box, choose where to save the test results, as described in "Running a Test to Check Your Application" on page 292, and click **OK**.

Updating a Test

When you update a test, QuickTest runs through the test to update the ActiveScreen and the expected values in checkpoints. This becomes the basis of comparison for subsequent test runs.

When QuickTest updates tests, it does not update parameterized values, such as data table data and environment variables. For information on parameterized tests and environment variables, see Chapter 10, “Parameterizing Tests.”

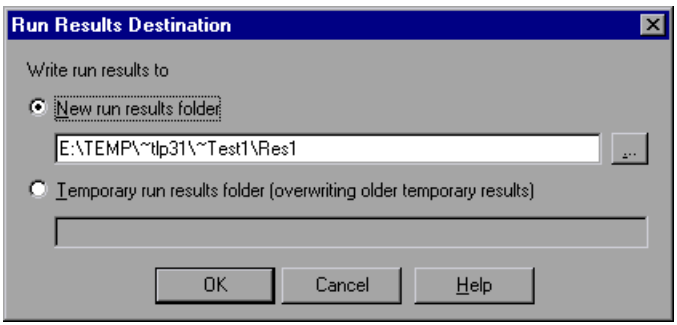
When QuickTest updates tests, it runs through one iteration of the test and one iteration of each action in the test. For information on actions, see Chapter 14, “Working with Actions.”

Note: Updating tests created in earlier versions of QuickTest may conserve disk space, as tests created in later versions of QuickTest use disk space more efficiently.

To run a test to update the expected results:

1 If your test is not already open, choose **File** > **Open** or click the **Open** button to open the test.

2 Choose **Test** > **Update Run**. The Run Results Destination dialog box opens with the New run results folder option selected by default. This option displays the default path and a folder name for the test run results.



3 To save the test run results under a different name, type it in the text box or click the browse button to locate the folder.

To run the test and overwrite the previous test run results, click **Temporary run results folder**.

Note: QuickTest stores temporary test run results for all tests in <*System Drive:**Temp**TempResults*>. The path in the text box of the **Temporary run results folder** option is read-only and cannot be changed.

---

4 Click **OK**. The Run Results Destination dialog box closes and QuickTest begins running the test. As QuickTest runs the test, it highlights each step in the test tree.

When the test stops running, the Test Results window opens unless you have cleared the **View results when test run ends** check box in the Run tab of the Options dialog box. The checkpoint node of the test results includes information on updated values of your checkpoint. For more information about the Options dialog box, see Chapter 24, "Setting Global Testing Options."

QuickTest runs the test and updates the ActiveScreen and the expected values in checkpoints.

Using Optional Steps

When running a test, if a step does not succeed in opening a particular dialog box, QuickTest does not necessarily interrupt the test run. It can bypass the step and continue to run the test. QuickTest will bypass any *optional* step. By default, QuickTest sets some dialog boxes as optional steps automatically. You can also set a step as optional.

Note: If you do not want to bypass dialog boxes using optional steps, you can use the Exception Editor to click a button, press Enter, or enter login information. For additional information, see Chapter 16, "Handling Unexpected Events and Errors."

Setting Optional Steps

When running a test, you can bypass certain steps if they are not required, by setting them as optional steps. For example, when recording a test, the site you are testing may prompt you to enter your user name and password in a login window. When you run the test, however, the site does not prompt you to enter your user name and password, because it has retained the information that was previously entered. In this case, the steps that were recorded for entering the login information are not required and should be marked as optional.

When running a test, if a step in an optional dialog box does not open, QuickTest automatically bypasses this step and continues to run the test. When the test run is completed, a message is displayed for the step that failed to open the dialog box, but the step does not cause the test to fail.

To set an optional step:

Right-click a step in the test tree and choose **Optional Step**. The Optional Step ? icon is added next to the selected step.

Note: You can also add an optional step from the Expert View by adding OptionalStep to the beginning of the VBScript statement. For example:

```
OptionalStep.Browser("browser_name").Page("page_name").Link("link_name")
```

For information on working in Expert View, see Chapter 22, "Testing in the Expert View." For information on the **OptionalStep** object, refer to the *QuickTest Object Model Reference*.

Default Optional Steps

QuickTest automatically considers steps that open the following dialog boxes as optional steps:

Logical Name	Title
Auto Complete	Auto Complete
File Download	File Download
IE message	Internet Explorer
Netscape message	Netscape
Password	Enter Network Password
Runtime error	Error
Security Alert	Security Alert
Security Information	Security Information
Security Warning	Security Warning
Username and Password Required	Username and Password Required

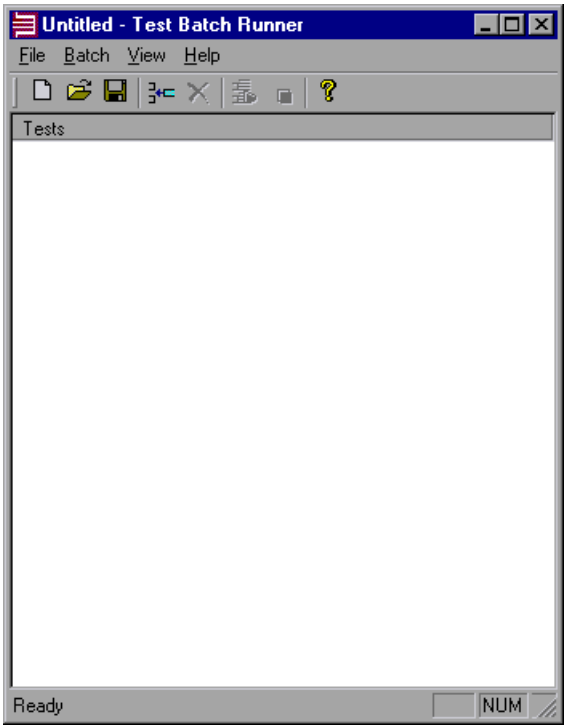
Running a Test Batch

You can use Test Batch Runner to run several tests in succession. The results for each test are stored in their default location. Using Test Batch Runner, you can set up a list of tests and save the list as an **.mtb* file so that you can easily run the same batch of tests again at another time. You can also choose to include or exclude a test in your batch list from running during a batch run.

Note: To enable Test Batch Runner to run tests, you must select the **Allow other Mercury tools to run tests** check box in the Run tab of the Options dialog box. For more information, see Chapter 24, "Setting Global Testing Options.".

To set up and run a test batch:

1 Choose **Programs** > **QuickTest** > **Tools** > **Test Batch Runner** from the Start menu. Test Batch Runner opens.



2 Click the **Add** button or choose **Batch** > **Add**. The Open Test dialog box opens.

3 Select a test you want to include in the test batch list and click **Open**. The test is added to the batch list.

4 Repeat step 3 for each test you want to include in the list. By default, the test is added to the bottom of the list.

To insert a test to another location in the list, select the test before which you would like to add the test and then add the test. The test is added above the selected test.

To remove a test from the list, click the **Remove** button , or choose **Batch** > **Remove**.

If you want to include a test in your list, but you do not want the test to be included in the next batch run, clear the check box next to the test name.

5 If you want to save your batch list, click the **Save** button , or choose **File** > **Save** or **File** > **Save As** and enter a name for your batch list. The file extension is *.mtb*.

6 When you are ready to run your test batch, click the **Run** button, or choose **Batch** > **Run**. If QuickTest is not already open, it opens, and the tests run sequentially. Once the batch run is complete, you can view the results for each test in its default test results folder (the *<test folder>\res#\report* folder).

For more information about Test Results, see Chapter 18, "Analyzing Test Results."

18

Analyzing Test Results

After you run a test, you can view a report of all the major events that occurred during the test run.

This chapter describes:

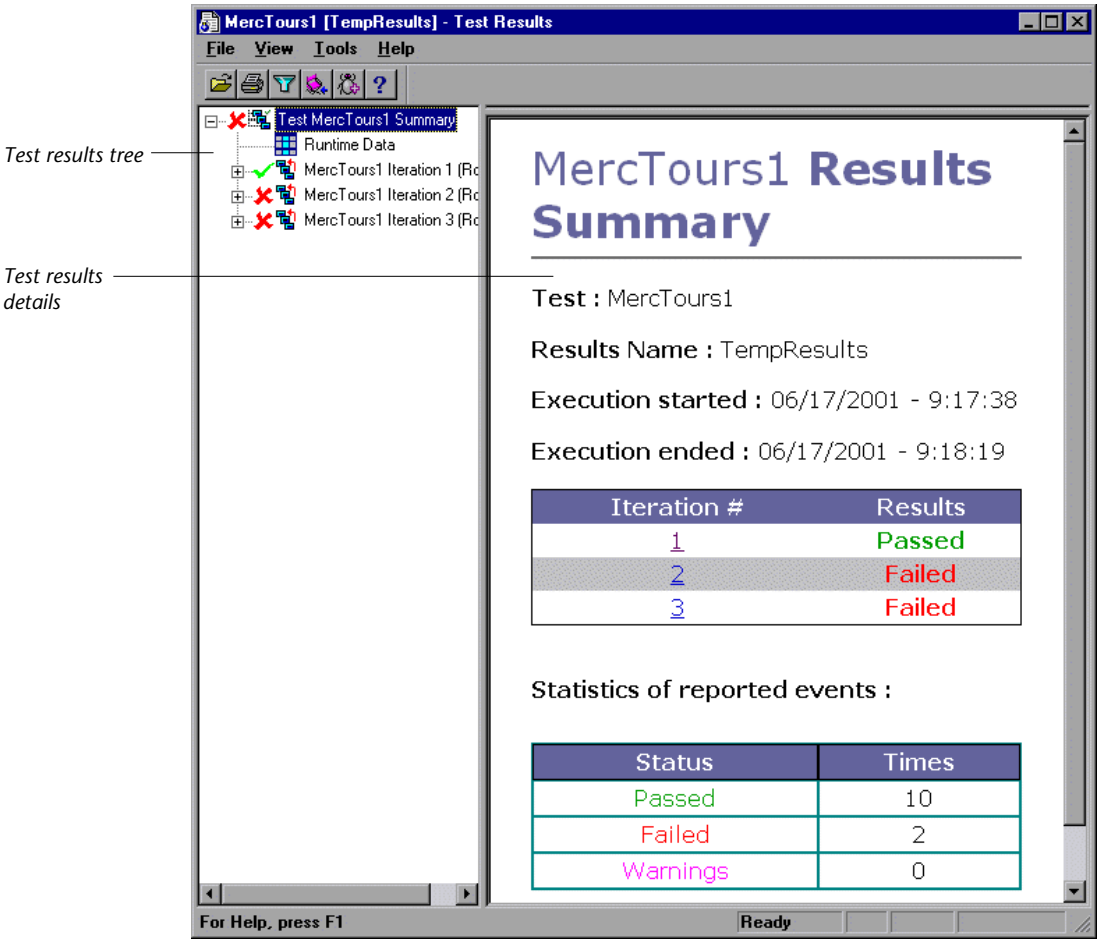
- The Test Results Window
- Viewing the Results of a Test Run
- Viewing the Results of Any Test Run
- Viewing the Results of a Checkpoint
- Viewing the Runtime Data Table
- Printing Test Results
- Reporting Defects Detected During a Test Run

About Analyzing Test Results

After you run your test, the test results are displayed in the Test Results window. This window contains a description of every step performed during the test run. If the test does not contain parameterized values, the Test Results window shows a single test iteration result. If the test does contain parameters, and the Run Properties for the action(s) and/or the test are set to run more than one iteration, the Test Results window shows a test iteration for each row in the table in the Data pane.

The Test Results Window

After a test run, you can view the results in the Test Results window. By default, the Test Results window automatically opens when a test run is completed. For more information on opening the Test Results window, see "Viewing the Results of a Test Run" on page 303.



Test Results Tree

The left pane in the Test Results window displays the *test results tree*—a graphical representation of the test results:

- The ✓ icon indicates a successful step.
- The ✗ icon indicates a failed step. Note that this causes all parent steps (up to the root action or test) to fail as well.
- The ! icon for a warning, which means that the step was unsuccessful but did not stop the test from running.

Note: The ! icon indicates a step that might have been expected to fail, such as an optional step. The ! ⊗ icon indicates a step that failed unexpectedly, such as when an object is not found for a checkpoint.

In the example above, the tree includes three iterations. The test results tree also includes the  icon that displays the *Runtime Data*—a table that shows the values used to run a parameterized test, or the output values retrieved from a test while it runs. Note that your test results are organized by action.

You can collapse or expand a branch in the test results tree in order to change the level of detail that the tree displays.

Test Results Details

The right portion of the window displays the *test results details*—additional information for a selected branch of the report tree.

By default, when the Test Results window opens, a test summary is displayed. It indicates the test name, the date and time of the test run, and whether a test iteration passed or failed.

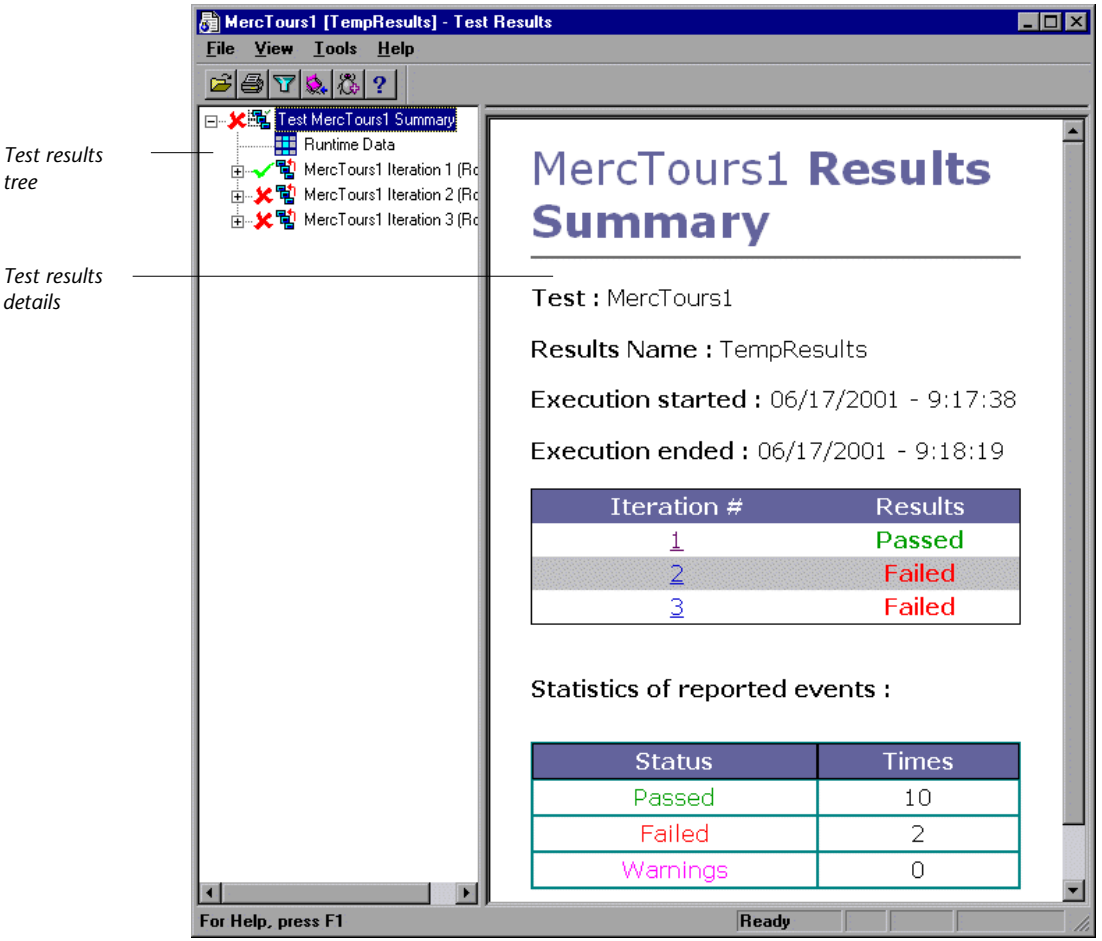
Viewing the Results of a Test Run

After a test run, you can view the results in the Test Results window. By default, the Test Results window automatically opens when a test run is completed. You can change this default setting in the Options dialog box. For more information, see “General Testing Options” on page 387.

To view the results of a test run:

1 If the Test Results window is not already open, click the **Test Results** button or choose **Test** > **Results**.

- If there are test results for the current test, they open in the Test Results window.
- If there are no test results for the current test, the Open Test Results dialog box opens. You can select the test results for any test, or you can search for the results file (*.qtp* extension) anywhere in the file system. Click **Open** to display the selected results in the **Test Results** window. For additional information on viewing test results for any test, see "Viewing the Results of Any Test Run" on page 308.



2 You can collapse or expand a branch in the test results tree in order to change the level of detail that the tree displays.

- To collapse a branch, click the Collapse (-) sign to the left of the branch icon. The report tree hides the details for the branch and the Collapse sign changes to Expand.
- To collapse all the branches in the report tree, choose **View** > **Collapse All**.

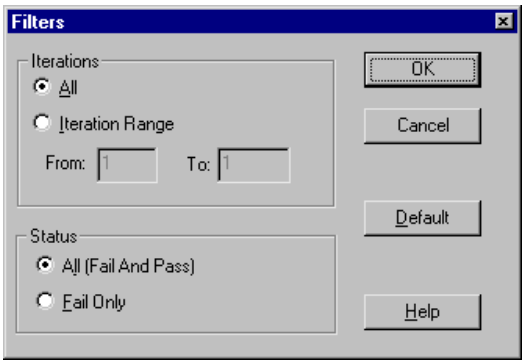
- To expand a branch, click the Expand (+) sign to the left of the branch icon. The tree displays the details for the branch and the Expand sign changes to Collapse.
- To expand all the branches in the report tree, choose **View** > **Expand All**.

3 You can view the results of an individual iteration, action, or step. The results can be one of three types:

- Iterations, Actions, and Steps that contain checkpoints are marked as **Passed** or **Failed** in the Test results details pane and are identified with the icon: ✔ or ✘.
- Iterations, Actions, and Steps that were run successfully, but do not contain checkpoints, are marked as **Done** in the Test results details pane.
- Steps that were not successful, but did not cause the test to stop running, are marked with a **Warning** in the Test results details pane and are identified with the icon: ! or !⊗.

Note: A test, iteration, or action containing a step with a **Warning** may still be marked as **Passed** or **Done**.

4 To filter the information contained in your test results report, click the **Filters** button or choose **View** > **Filters**. The Filters dialog box opens.



The default filter options are displayed above.

- Click **Iteration Range** to limit the test results to a specified range of test iterations.
- Click **Fail Only** to limit the test results to test iterations that failed.

5 To view other test run results, click the **Open** button or choose **File** > **Open.** For additional information, see "Viewing the Results of Any Test Run" on page 308.

6 To print the test results, click the **Print** button or choose **File** > **Print**. For additional information, see "Printing Test Results" on page 312.

Note: If you have TestDirector installed, you can add a defect to a TestDirector project. For additional information, see "Reporting Defects Detected During a Test Run," on page 313.

---

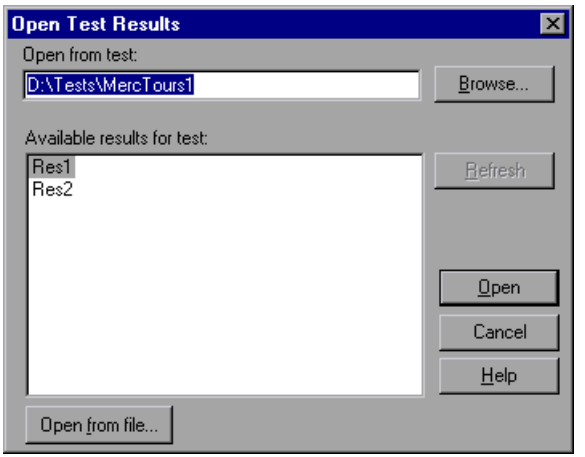
7 Choose **File** > **Exit** to close the Test Results window.

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** > **Astra QuickTest** > **Test Results Viewer**.

Viewing the Results of Any Test Run

You can open and view the results for any test.

If you choose **Test** > **Results** when there are no results for the current test, or if you choose **File** > **Open** from within the Test Results window, the Open Test Results dialog box opens.



You can view:

- any results for the current test
- the results for any test (search by test)
- any test results (search in the file system)

To view any results for the current test:

1. Highlight the results to view in the **Available results for test** box.
2. Click **Open**.

Tip: To update this list, click **Refresh**.

To view the results for any test (search by test):

1. Click **Browse** to open the Open Test dialog box.
2. If necessary, browse to the folder containing the desired test.
3. Highlight the test whose results you want to view and click **Open**.
4. Highlight the results to view in the **Available results for test** box and click **Open**.

To view any test results (search in the file system):

1. Click **Open From File** to open the Select Results File dialog box.
2. Browse to the folder where the test results are stored.
3. Highlight the desired test results file and click open.

Tip: Test result files have the **.qtp* extension.

Note: By default, results are stored in the *[TestName]\[ResultsName]\Report* folder.

Viewing the Results of a Checkpoint

By adding checkpoints to your tests, you can compare pages, text strings, objects, and tables in different versions of your Web site. This enables you to ensure that your Web site functions as desired.

When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails. You can view the results of the checkpoint in the Test Results window.

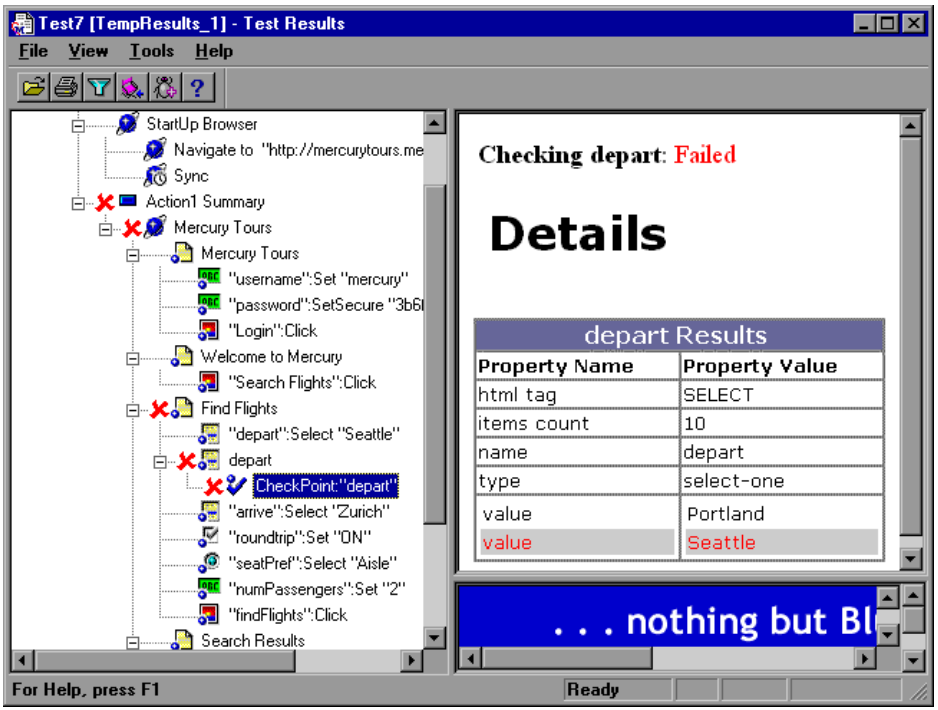
For more information on checkpoints, see Chapter 6, “Creating Checkpoints.”

To view the results of a checkpoint:

1 If the Test Results window is not already open, then click the **Test Results** button or choose **Test** > **Results**. If you have more than one set of results, or no results, the Open Test Results dialog box opens. Select a results file (*.qtp* extension). Click **Open**. The Test Results window opens. If you have only one set of results for your test, the Test Results window opens directly.

2 In the left pane of the Test Results window, expand the branches of a test iteration.

3 Click a checkpoint branch. The right pane displays detailed results of the selected checkpoint.



In the above example, the detailed results of the failed checkpoint indicates that the expected results and the current results do not match. The expected value of the flight departure is "Portland," but the actual value is "Seattle."

4 Choose **File** > **Exit** to close the Test Results window.

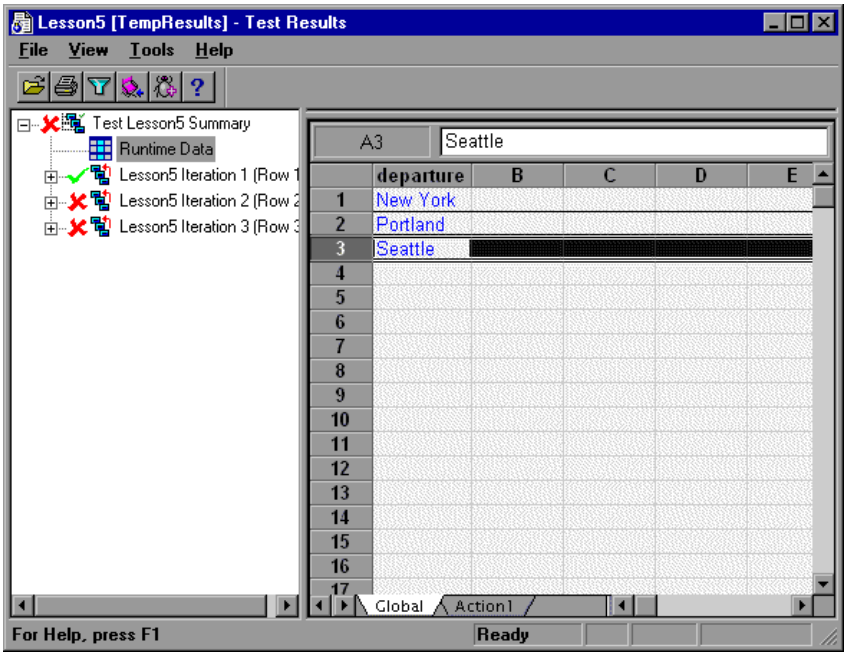
Viewing the Runtime Data Table

After you run a parameterized test, the Runtime data table displays the values used to run a parameterized test, or the output values retrieved from a test while it runs. For more information on parameterization, see Chapter 10, “Parameterizing Tests.” For more information on output values, see Chapter 11, “Creating Output Values.” For more information on the test Data Table, see Chapter 15, “Working with Data Tables.”

To view the Runtime data table:

1 If the Test Results window is not already open, then click the **Test Results** button or choose **Test** > **Results**. If you have more than one set of results, or no results, the Open Test Results dialog box opens. Select a results file (*.qtp* extension). Click **Open**. The Test Results window opens. If you have only one set of results for your test, the Test Results window opens directly.

2 In the left pane of the Test Results window, highlight the **Runtime Data** icon. The right pane displays the Runtime data table.



In the above example, the Runtime data table contains the parameterized flight departure values.

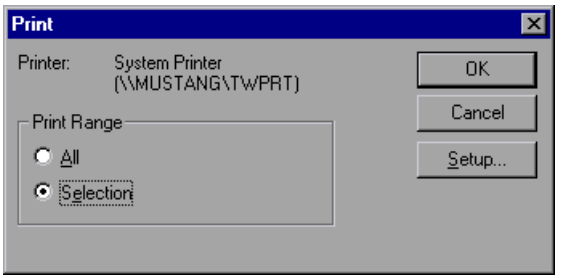
3 Choose **File** > **Exit** to close the Test Results window.

Printing Test Results

You can print your test results from the Test Results window.

To print the test results:

1 To print the report, click the **Print** button or choose **File** > **Print**. The Print dialog box opens.



2 Select a **Print Range** option:

- ➤ Select **All** to print the entire results report.
- ➤ Select **Selection** to only print a selected branch in the report tree.

3 Click **OK** to print.

Reporting Defects Detected During a Test Run

If a test run detects a defect, you can report it to a TestDirector project directly from the Test Results window.

To report a defect to TestDirector:

1 Choose **Tools** > **TestDirector Connection** or click the **TestDirector Connection** button to connect to a TestDirector project. For additional information, see Chapter 28, "Working with TestDirector."

2 Choose **Tools** > **Add Defect** or click the **Add Defect** button to open the Add Defect dialog box in the specified TestDirector project. For additional information, refer to the *TestDirector User's Guide*.

19

Debugging Tests

Controlling test runs can help you to identify and eliminate defects in your tests.

This chapter describes:

- Using the Step Commands
- Pausing Test Runs
- Setting Breakpoints
- Deleting Breakpoints
- Using the Debugger Views
- Example of Debugging a Test

About Debugging Tests

After you create a test you should check that it runs smoothly, without errors in syntax or logic. In order to detect and isolate defects in a test, you can use the Step and Pause commands to control how it runs. In addition, you can also control how the test runs by setting breakpoints. When you stop the test at a breakpoint, you can use the Debugger View to check or modify the current value of VBScript objects and variables in your test.

The following Step commands are available:

- The Step Into command calls a function or displays another test.
- The Step Out command—used in conjunction with Step Into—completes the execution of a function or called test.
- The Step Over command executes a function or a called test.

Tip: You can use the **Run from Line** feature to debug your test. This runs your test from the cursor to the end of the current action. For additional information, see "Running a Test to Debug It" on page 294.

You can also use the Pause command to temporarily suspend a test run. When you resume running the test, it continues from the point where you invoked the Pause command.

In addition, you can also control test runs by setting breakpoints and viewing the Debugger Views. A breakpoint pauses a test run at a pre-determined point, enabling you to examine the effects of specific steps.

Using the Step Commands

You can run a single line of a test using the Step Into, Step Out, and Step Over commands.

Step Into

Choose **Debug** > **Step Into** or click the **Step Into** button to run only the current line of the active test. If the current line of the active test calls another test or a function, the called test or function is executed in its entirety, and the called test or function is displayed in the QuickTest window.

Step Out

Choose **Debug** > **Step Out** or click the **Step Out** button only after entering a test or a user-defined function using Step Into. Step Out runs to the end of the called test or user-defined function, returns to the calling test, and then pauses the test run.

Step Over

Choose **Debug** > **Step Over** or click the **Step Over** button to run only the current step in the active test. When the current step calls another test or a user-defined function, the called test or function is executed in its entirety, but the called test script is not displayed in the QuickTest window.

Pausing Test Runs

You can temporarily suspend test runs by choosing **Debug** > **Pause** or clicking the **Pause** button. A paused test stops running when all previously interpreted steps have been run.

To resume running a paused test, click the **Run** button or choose **Test** > **Run**. The test run continues from the point that you invoked the Pause command.

Setting Breakpoints

By setting a breakpoint you can stop a test run at a specific place in a test. The test pauses before executing the selected line. A breakpoint is indicated by a red-colored hand in the left margin of the test window. QuickTest pauses the test run when it reaches a breakpoint. You can examine the effects of the test run up to the breakpoint, make any necessary changes, and then continue running the test from the breakpoint.

You can use breakpoints to:

- suspend a test run and inspect the state of your site
- mark a point from which to begin stepping through a test using the Step commands

To set a breakpoint:

1 Click a step or a line in the test where you want the test run to stop.

2 Choose **Debug** > **Toggle Breakpoint** or click the **Toggle Breakpoint** button. The breakpoint symbol is displayed in the left margin of the QuickTest window.

Note: The breakpoints you define are active only during your current QuickTest session. If you terminate your QuickTest session, you must redefine breakpoints to continue debugging the test in another session.

Deleting Breakpoints

You can delete a single breakpoint or all breakpoints defined for the current test using the Debug menu.

➤ To delete a single breakpoint, click a line in your test with the breakpoint symbol and choose **Debug** > **Insert/Remove Breakpoint** or click the **Insert/Remove Breakpoint** button.

The breakpoint symbol is removed from the left margin of the QuickTest window.

➤ To delete all breakpoints, choose **Debug** > **Clear All Breakpoints** or click the **Clear All Breakpoints** button.

All breakpoint symbols are removed from the left margin of the QuickTest window.

Using the Debugger Views

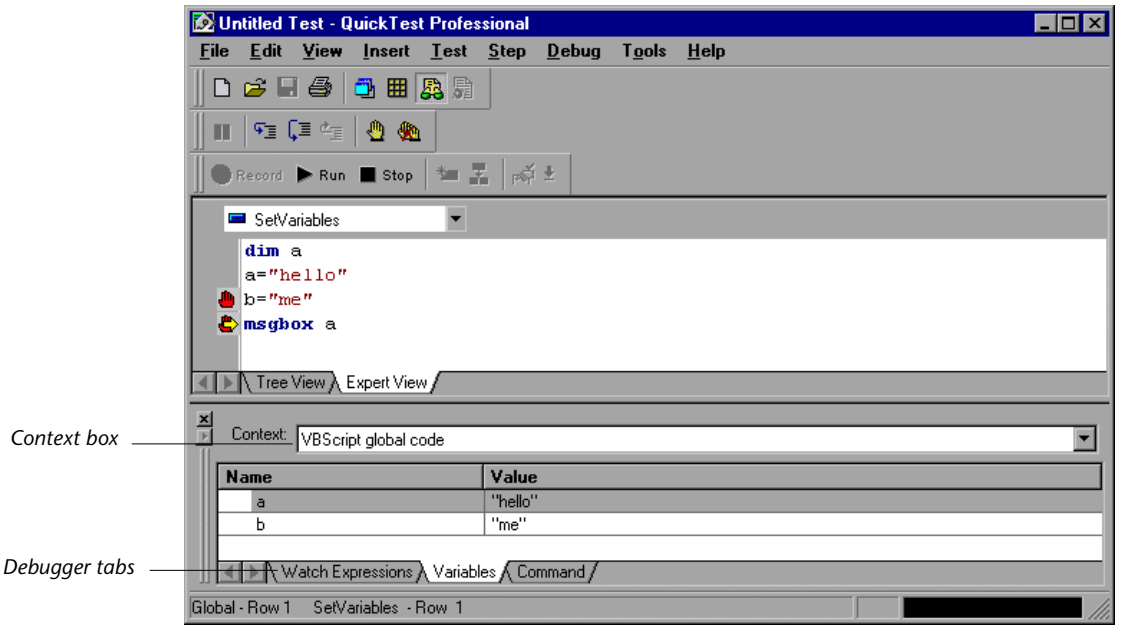
When a test stops at a breakpoint, you can use the Debugger pane to view, set, or modify the current value of objects or variables in your test.

To open the Debugger pane:

1 Run a test with one or more breakpoints.

2 When the test pauses at the first breakpoint, choose **View** > **Debug View** or click the Debugger Views button. The Debugger pane opens at the bottom of the QuickTest screen. If the Data pane is also open, the Debugger pane opens on the bottom right of the screen.

If the Data pane is not open, the Debugger pane spreads across the bottom of the QuickTest window.



The Debugger tabs can display the values of variables or objects in the main script of the current action or in a selected subroutine. To switch between the main script of the action (VBScript: global code) and the subroutines and functions of the action, choose the script you want from the **Context** box.

Watch Expressions Tab

Use the Watch Expressions tab to view the current value of any variable or VBScript object that you enter in the Watch Expressions table. 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. If the value of the object or variable changes when you continue to run the test, the value in the Watch Expressions tab is updated.

Note: QuickTest updates the value of the object or variable in the Watch Expressions tab when running a test step by step.

Variables Tab

Use the Variables tab to view the current value of all variables in the current action (or selected subroutine) that have been identified up to the point where the test stopped. If the value of a variable changes when you continue to run the test, the value in the Variables tab is updated.

Note: QuickTest updates the value of the variable in the Variables tab when running a test step by step.

Command Tab

Use the Command tab enables to execute a line of script in order to set or modify the current value of a variable or VBScript object in your test. When you continue running the test, QuickTest uses the new value that was set in the command.

Example of Debugging a Test

Suppose you create an action in your test that defines variables that will be used in other parts of your test. You can add breakpoints to the action to see how the value of the variables change as you run the test. You can also change the value of one of the variables during a breakpoint to see how the test handles the new value.

Step 1: Create the New Action

Open a test and insert a new action called "SetVariables". For more information about inserting actions see Chapter 14, "Working with Actions."

Enter the VBScript code for the action in the Expert View as follows:

```
Dim a
a="hello"
b="me"
MsgBox a
```

For more information about the Expert View, see Chapter 22, "Testing in the Expert View."

Step 2: Add Breakpoints

Add a breakpoint at line 3 and line 4. For more information about adding breakpoints, see "Setting Breakpoints," on page 317.

Step 3: Begin Running the Test

Run the test. The test stops at the first breakpoint.

Step 4: Check the Value of the Variables in the Debugger Pane

Choose **View** > **Debug View** to open the Debugger pane.

Select the **Watch Expressions** tab on the Debugger pane. In the first cell in the Name column, type "a" (without quotes) and press **Enter** on the keypad. The Value column indicates that the a variable is currently hello, because the breakpoint stopped after the value of variable a was initiated. In the next cell of the Name column, type "b" (without quotes) and press **Enter** on the

keypad. The Value column indicates that Variable b is undefined, because the test stopped before variable b was declared.

Note: The breakpoint causes the test to stop before executing the corresponding line.

Select the **Variables** tab in the Debugger pane. Note that the variable a is displayed with the value hello, because a is the only variable that has been initiated at this point in the test.

Step 5: Check the Value of the Variables at the Next Breakpoint

Click the **Run** button to continue running the test. The test stops at the next breakpoint. Note that the values of variables a and b have both been updated in the Watch Expressions and Variables tabs.

Step 6: Modify the Value of a Variable Using the Command Tab

Select the **Command** tab in the Debugger pane. Type: a="This is the new value of a" at the command prompt, and press **Enter** on the keypad. Click the **Run** button to continue running the test. The message box that appears displays the new value of 'a'.

Part V

Advanced Features

20

Configuring Event Recording

If QuickTest does not record all the events you need, you can configure the events you want to record for each type of Web object.

This chapter describes:

- Selecting a Standard Event Recording Configuration
- Customizing the Event Recording Configuration
- Saving and Loading Custom Configuration Files
- Resetting Event Recording Configuration Settings

About Configuring Event Recording

QuickTest creates your test by recording the *events* you perform on your Web-based application. An event is a notification that occurs in response to an action, 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 a 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 only if they are connected to a behavior.

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

Changing the event configuration settings does not affect tests 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 test that is affected by the change.

Selecting a Standard Event Recording Configuration

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.

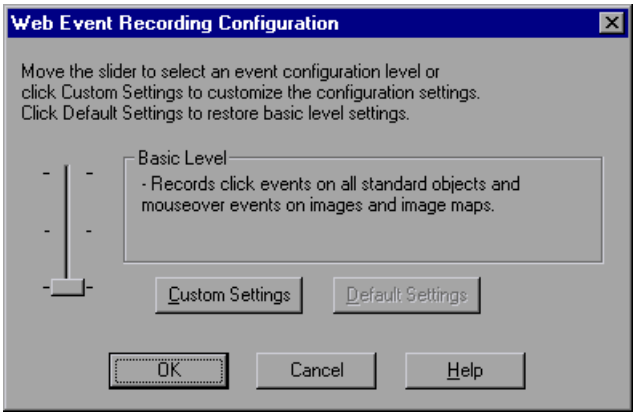
The Web Event Recording Configuration dialog box offers three standard event configuration levels.

Level	Description
Basic	**Default** • Always records click events on standard Web objects. • Always records the submit event within forms. • Records click events on other objects with a handler or behavior connected. • Records the event following a mouseover event on images and image maps.

Level	Description
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 333.

To set a standard event recording configuration:

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



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

3 Click **OK**.

Customizing the Event Recording Configuration

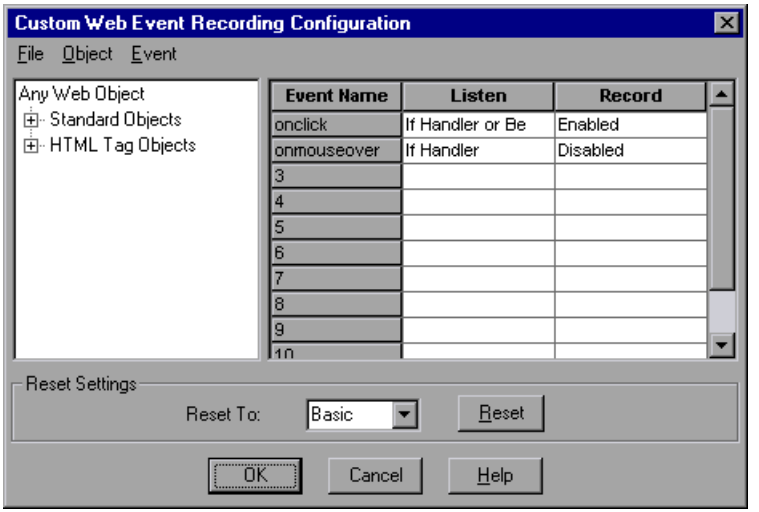
If the standard event configuration levels do not exactly match your recording needs, you can customize the event recording configuration using the Custom Web Event Recording Configuration dialog box.

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 for all objects
- add or delete events for which QuickTest should listen for one or more specific objects
- modify the listening or recording settings of an event for which QuickTest listens for all objects
- modify the listening or recording settings of an event for one or more specific objects

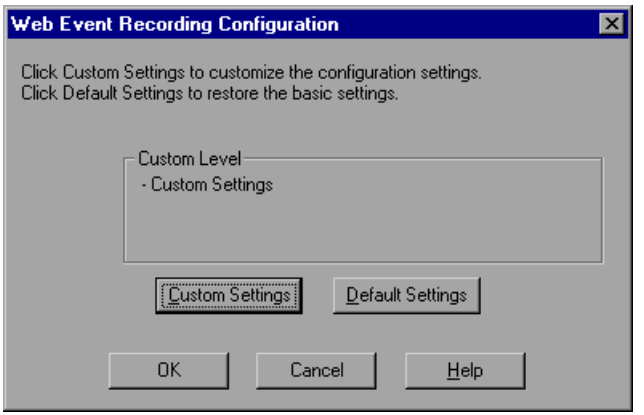
To customize the event recording configuration:

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.



3. Set the event recording configuration options you want. The sections below describe the event recording configuration options in detail.

4 Click **OK**. The Custom Web Event Recording Configuration dialog box closes. The slider on the Web Event Recording Configuration dialog box disappears and the configuration description displays: Custom Settings.



Adding and Deleting Objects in the Custom Configuration Object List

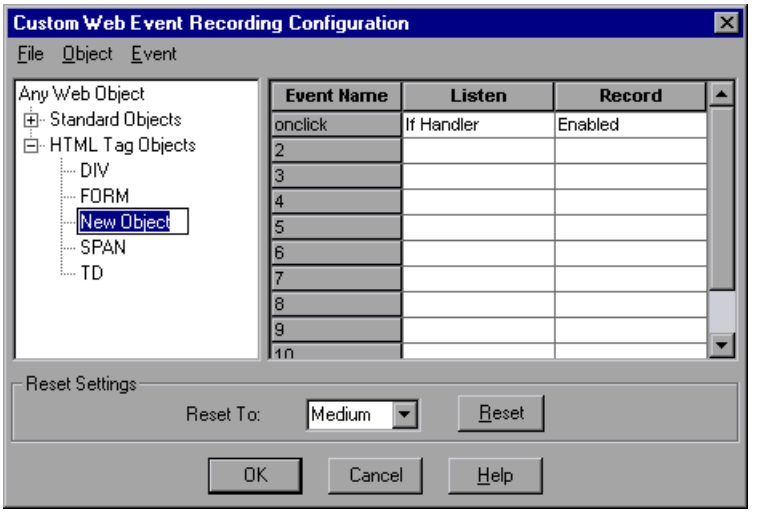
The Custom Web Event Recording Configuration dialog box lists objects in an object hierarchy. The top of the hierarchy is Any Web Object. The settings for Any Web Object apply to any object on the Web page being tested, for which there is no specific settings. Below this are the Standard and HTML Tag 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.
- QuickTest always listens to the objects listed under standard objects. Therefore, you cannot delete or add to the objects in this category.
- 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 From the Custom Web Event Recording Configuration dialog box, choose **Object** > **Add**. A "New Object" object is displayed in the HTML Tag Objects list.



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

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

For more information on adding or deleting events, see "Adding and Deleting Listening Events for an Object," on page 332.

For more information on listening and recording settings, see "Modifying the Listening and Recording Settings for an Event," on page 333.

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 standard objects category.

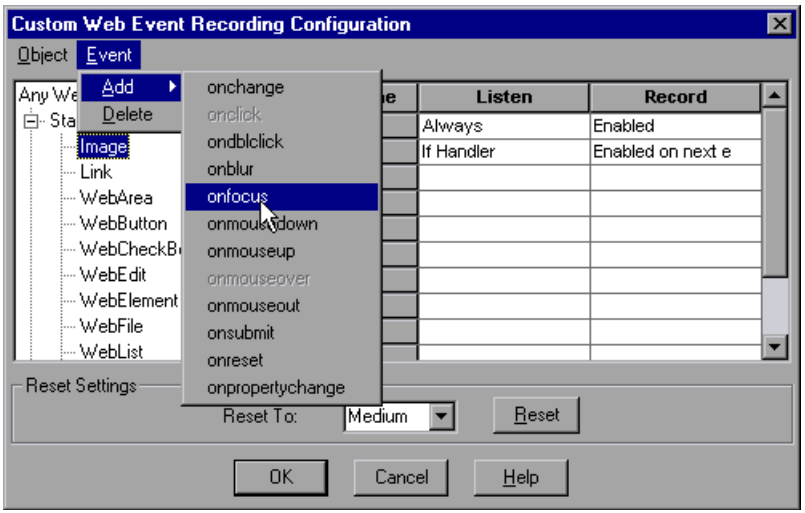
Adding and Deleting Listening Events for an Object

You can modify the list of events that trigger QuickTest to listen to an object.

To add listening events for an object:

1 From the Custom Web Event Recording Configuration dialog box, select the object to which you want to add the 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, the event is set to record when a handler is attached to the object.

For more information on listening and recording settings, see "Modifying the Listening and Recording Settings for an Event," on page 333.

To delete listening events for an object:

1 From 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.

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 and/or if a DHTML behavior is attached to the event.

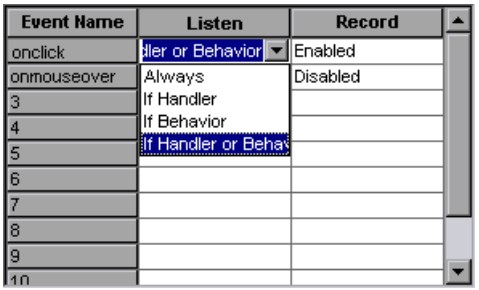
An event *handler* is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs.

A DHTML *behavior* is a simple, lightweight component that 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.



You can select **Always, If Handler**, **If Behavior**, or **If Handler or Behavior**.

Recording Status

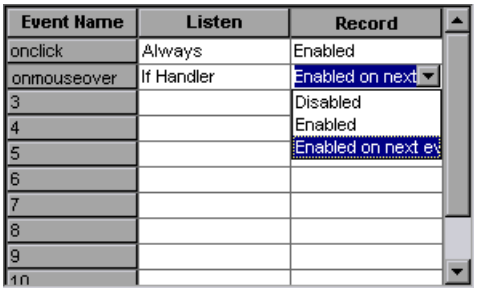
For each event to which QuickTest listens, you can enable recording, disable recording, or enable recording only if the next event is dependant on this event.

- **Enabled** - records the event each time the listening criterion is met.
- **Disabled** - does not record the specified event and ignores event *bubbling* where applicable.

  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.
- **Enabled on next event** - records the event only if the subsequent event occurs on the same object and is dependant on this event. For example, suppose a mouseover behavior modifies an image link. You may not want to record the mouseover event each time you happen to move the mouse over this image. Because only the image that is displayed after the mouseover event enables the link event, however, it is essential that the mouseover event is recorded before a click event on the same object. This option applies only to the Image and WebArea standard 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.



Saving and Loading Custom Configuration Files

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

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 Event Recording Configuration,” on page 328.
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 configuration file, and enter a configuration file name. The extension for configuration files is *.xmd*.
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 (**.xmd*) that you want to load and click **Open**. The dialog box closes and the selected configuration is loaded.

Resetting Event Recording Configuration Settings

You can restore standard settings after you have set Custom settings by resetting the event recording configuration settings to the basic level from the 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. For more information, see "Saving and Loading Custom Configuration Files," on page 335.

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 redisplayed 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 326.

You can 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.

21

Enhancing Your Tests with Programming

After recording a test, you can use QuickTest to enhance your test using a few simple programming techniques.

This chapter describes:

- Inserting Methods
- Using Conditional Statements
- Sending Messages to Your Test Results
- Adding Comments

About Enhancing Your Tests with Programming

When recording, a test is generated by recording the typical processes that you perform on your Web site. As you navigate through your site, QuickTest graphically displays each *step* you perform as an icon in a *test tree*.

Once you record your test, you can increase its power and flexibility by programming. QuickTest includes the *Method Wizard*, a programming tool that helps you to quickly and easily add recordable and non-recordable methods to your test. You can use the wizard to add methods that perform operations on Web objects or retrieve information from your site. For example, you can add a step that checks that an object exists, or you can retrieve the return value of a method.

QuickTest also enables you to incorporate decision-making into your test. You can add conditional statements to control the logical flow of your test.

In addition, you can define messages in your test that QuickTest sends to your test results. To improve the readability of your tests, you can also add comments to your test.

This chapter introduces some programming concepts and shows you how to use simple programming techniques in the Tree View in order to create more powerful tests. For information on how to use programming concepts in the Expert View, see Chapter 22, "Testing in the Expert View."

Inserting Methods

After recording, you can add additional methods to your tests using the Method wizard. With the wizard you can add recordable and non-recordable methods that perform operations on objects or retrieve information from your site. For example, the **GetROProperty** method enables you to query the method argument value. You can use the return value of the method as an output value or as part of a conditional statement.

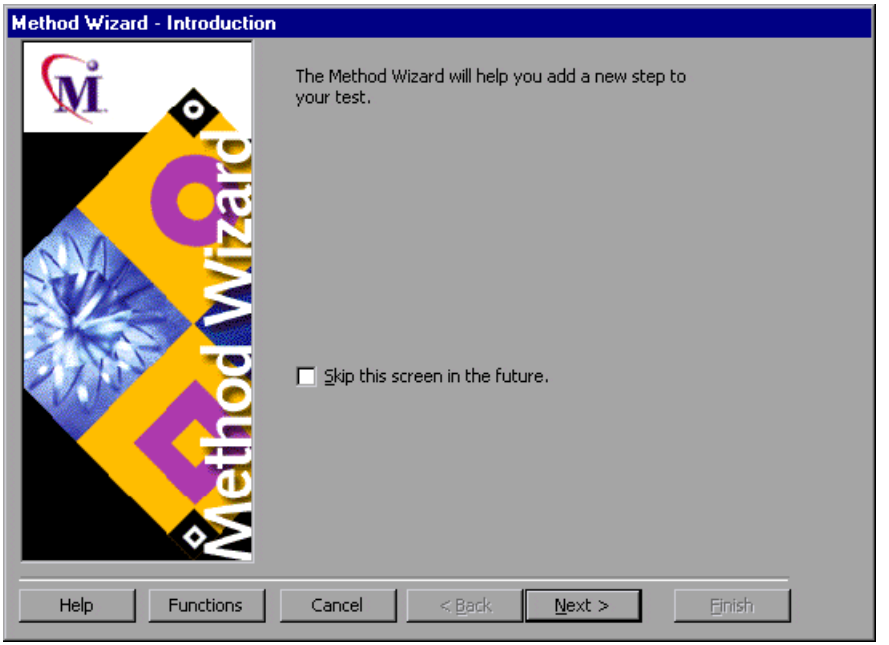
To insert a method in a test:

1 In the Tree View, click a step in the test tree and choose **Insert** > **Step** > **Method** or right-click the step and choose **Insert Method**.

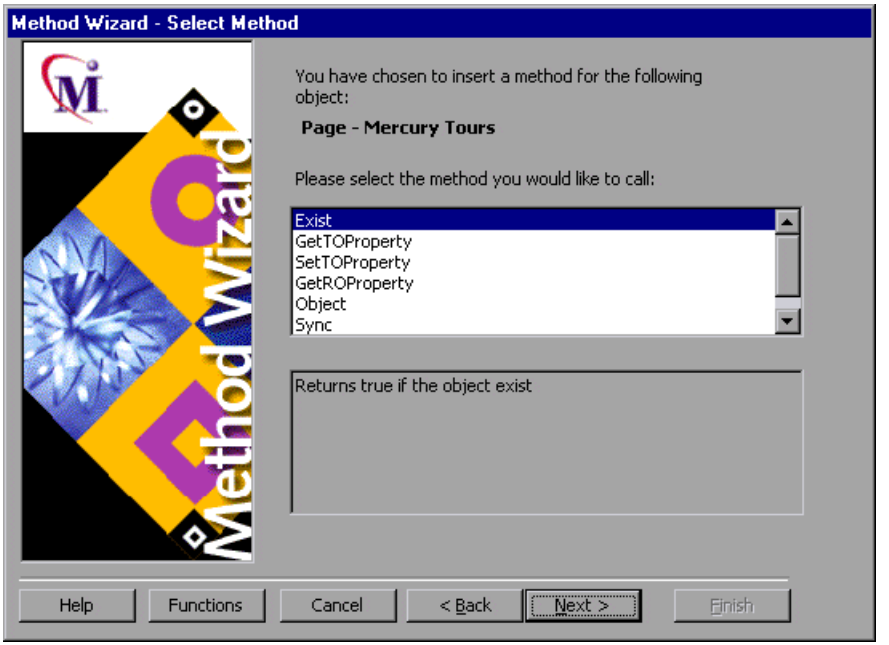
Tip: To add a method from the Expert View, click a statement in the test script. Right-click the highlighted object in the ActiveScreen and choose **Insert Method**. The **Object Selection - Object Method Wizard** opens. Select an object and click **OK**.

---

2 The Method Wizard - Introduction screen opens. Click **Next**.

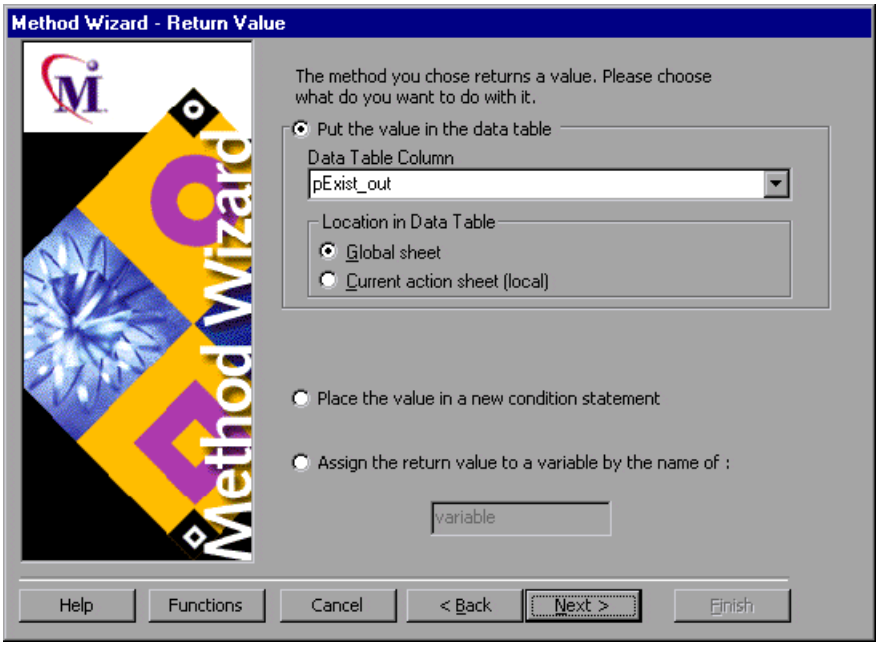


3 The Method Wizard - Select Method screen opens.



Select a method and click **Next**.

4 If the method you chose returns a value, the Method Wizard - Return Value screen opens. Otherwise, proceed to the next step.



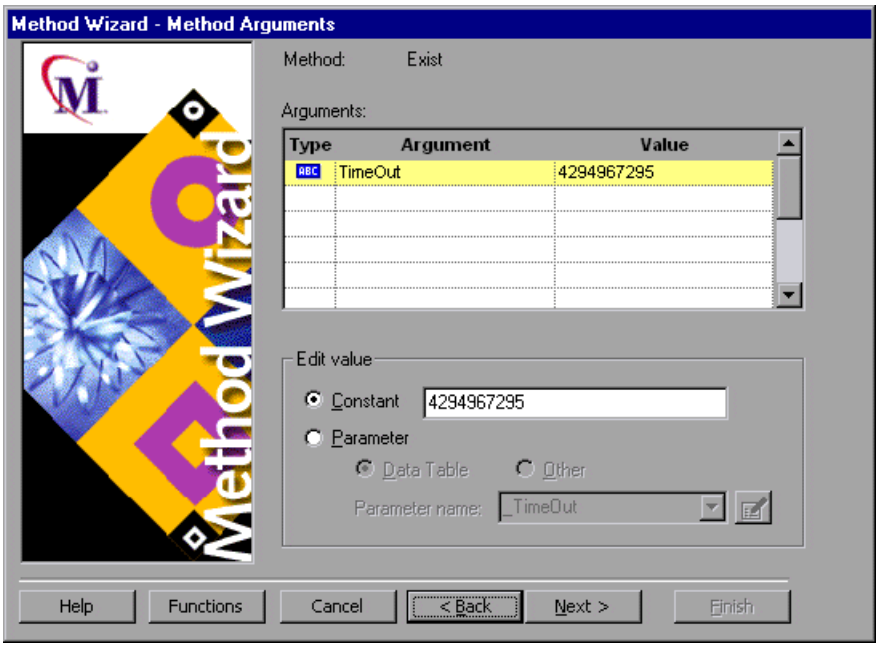
The following options are available:

Option	Description
Put the value in the data table (default)	Inserts the return value of the method as an output value into your Data pane. For more information, see Chapter 11, “Creating Output Values.”
Data Table Column	Sets the name for the output value. You can accept the default name, select from the list, or enter a new name. For more information, see Chapter 11, “Creating Output Values.”
Location in Data Table	Sets the data as global or action data in the Data pane.

Option	Description
Place the value in a new condition statement	Inserts the return value of the method into a conditional statement. For more information, see "Using Conditional Statements" on page 349.
Assign the return value to a variable by the name of:	Assigns the return value to a variable of the specified name.

Click **Next** to continue.

5 If the method you chose has arguments, the Method Wizard - Method Arguments screen opens. Otherwise, proceed to the next step.



The screen displays the arguments you can parameterize, in a pane listing the arguments, their values, and their types:

Option	Description
Type	The  icon indicates that the argument value is currently a constant. The  icon indicates that the argument value is currently a data table parameter. The  icon indicates that the argument value is currently an environment variable parameter. The  icon indicates that the argument value is currently a random number parameter.
Argument	The name of the argument whose value will be parameterized.
Value	The value of the argument to parameterize.

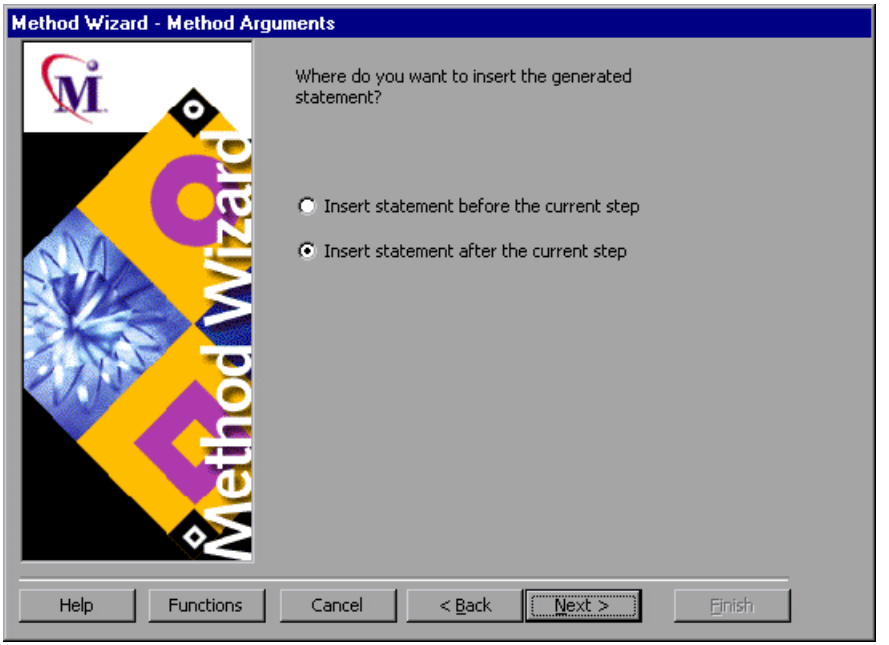
In the **Edit value** section, you use the following options to edit the argument value.

Option	Description
Constant (default)	Sets the expected value of the argument as a constant.
Parameter	Sets the expected value of the argument as a parameter. For more information, see Chapter 10, “Parameterizing Tests.”
Data Table	Specifies the parameter as a Data Table parameter. The value of the argument is determined by the data in the Data pane for each iteration of the test run. For more information about creating Data Table parameters, see “Data Table Parameters,” on page 152.
Parameter name	Specifies the name of the parameter in the Data Pane. **Note:** The **Parameter name** box is displayed only when the Data Table option is selected.

Option	Description
Other	Specifies that the parameter is a Random Number parameter or an Environment Variable parameter. The text box displays the name of the parameter. For more information about creating Random Number parameters, see "Random Number Parameters," on page 163. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156.
Edit parameter options 	If you select the **Data Table** option, the Edit parameter options button opens the DataTable Parameter Options dialog box, where you can also specify the parameter name and set the parameter as global or current action data. For more information, see "Data Table Parameters," on page 152. For more information, see Chapter 14, "Working with Actions." If you select the **Other** option, the Edit parameter options button opens the Parameter Options dialog box, where you can specify your parameter as a Random Number type or an Environment Variable type.

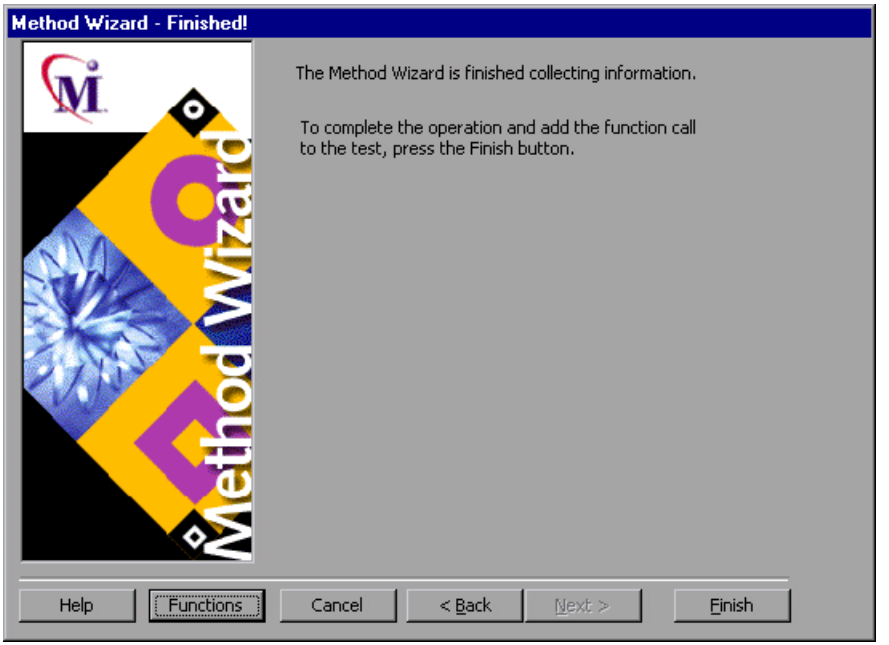
Click **Next** to continue.

6 The Method Wizard - Method Arguments screen for Insert Statement opens.



Select to insert the method before or after the current step in the test tree and click **Next**.

7 The Method Wizard - Finished screen opens.



Click **Finish** to complete the process and add the method to your test.

A tree item with a method icon is added to your test tree.

Using Conditional Statements

You can control the flow of your test with conditional statements. Using conditional statements, you can incorporate decision-making into your tests using *If...Then...Else* statements.

The *If...Then...Else* statement is used to evaluate whether a condition is true or false and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. The following comparison operators are available: *less than* <, *less than or equal to* <=, *greater than* >, *greater than or equal to* >=, *not equal* <>, *and equal* =.

Your *If...Then...Else* statement can be nested to as many levels as you need. It has the following syntax:

If *condition* **Then** *statements* [**Else** elsestatements] **End If**

Or, you can use the block form syntax:

If *condition* **Then**
    [*statements*]
  [**ElseIf** *condition-n* **Then**
    [*elseifstatements*] . . .
  [**End If**]
[**Else**
    [*elsestatements*]
End If

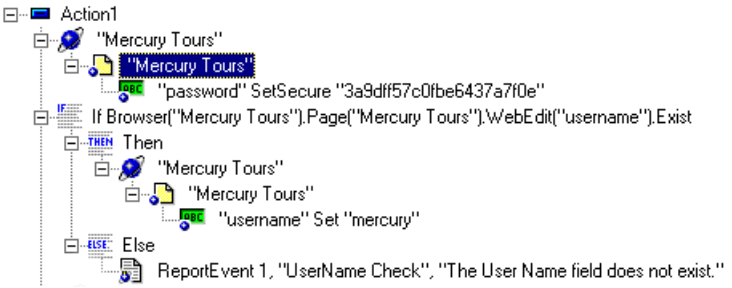
Part of Statement	Description
condition	One or more expressions that evaluate to true or false. If the condition is null, it is treated as false.
statements	One or more statements, separated by colons, that are executed if the condition is true.
condition-n	Same as *condition*.
elseif statements	One or more statements, separated by colons, that are executed if the associated *condition-n* is true.

Part of Statement	Description
else statements	One or more statements, separated by colons, that are executed if no previous *condition-n* expression is true.
End If statement	Terminates each **If** statement.

For example, the statement below (as it is displayed in the Expert View) checks that the user name edit box exists in the Mercury Tours site. **If** the edit box exists, **then** a user name is entered; **else** a message is sent to test results.

```
If Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Exist Then
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set "mercury"
Else
Reporter.ReportEvent 1, "UserName Check", "The User Name field does not exist."
End If
```

The same example is displayed in the Tree View as follows:



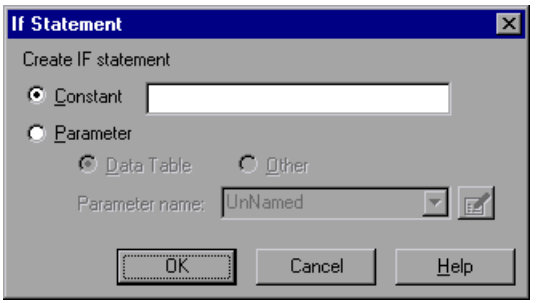
In the test tree, the following icons are used to indicate the different levels of *If...Then...Else* statements:

Icon	Description
	Starts an *If* statement.
	Starts a *Then* statement.
	Starts an *Elseif* statement.
	Starts an *Else* statement.

To add a conditional statement:

1 In the Tree View, click a step in the test tree.

2 Choose **Insert** > **Step** > **Conditional Statement** > **If...Then**. The If Statement dialog box opens.



3 Set the expression as a constant or a parameter. Note that the expression must be a Boolean expression.

➤ To set the expression as a constant, select **Constant**, and type the expression in the box. For example, type i>5.

➤ To set the expression as a parameter, select **Parameter** and choose one of the options below. For more information, see Chapter 10, "Parameterizing Tests."

- To set the expression as a data table parameter, select **Data Table** and choose a parameter from the **Parameter name** list or enter a new parameter. The value of the expression is determined by the data in the Data pane for each iteration of the test run. For more information about creating Data Table parameters, see "Data Table Parameters," on page 152.

  Click the **Edit parameter options** button  to open the **DataTable Parameter Options** dialog box, where you can also specify the parameter name and set the parameter as global or current action data. For more information, see Chapter 14, "Working with Actions."

- To set the expression as a Random Number parameter or an Environment Variable parameter, select **Other**. For more information about creating Environment Variable parameters, see "Environment Variable Parameters," on page 156. For more information about

creating Random Number parameters, see "Random Number Parameters," on page 163.

Click the **Edit parameter options** button  to open the **Parameter Options** dialog box, where you can specify your parameter as a Random Number type or an Environment Variable type.

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

5 To complete the **Then** statement you can:

- ➤ Record a new step, and then use the Cut/Paste commands to add it to your **Then** statement.
- ➤ Copy an existing step and paste it in your **Then** statement.
- ➤ Click and drag a step to move it to your **Then** statement.

6 To nest an additional level to your statement, click the **Then** statement and choose one of the following options:

To add:	Choose:
an **If** statement	**Insert** > **Step** > **Conditional Statement** > **If...Then**
an **Elseif** statement	**Insert** > **Step** > **Conditional Statement** > **Elseif...Then**
an **Else** statement	**Insert** > **Step** > **Conditional Statement** > **Else**

To complete the new statement you can:

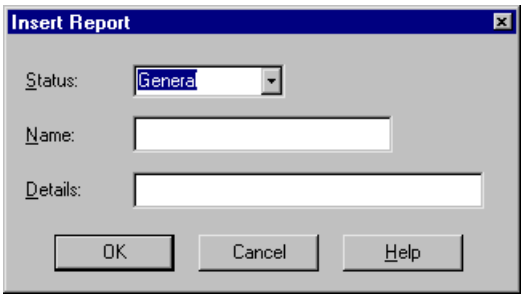
- ➤ Record a new step, and then use the Cut/Paste commands to add it to your statement.
- ➤ Copy an existing step and paste it in your statement.
- ➤ Click and drag a step to move it to your statement.

Sending Messages to Your Test Results

You can define a message in your test that QuickTest sends to your test results. For example, suppose you want to check that a password edit box exists in the Mercury Tours site. If the edit box exists, then a password is entered. Otherwise, QuickTest sends a message to the test results indicating that the object is not found.

To send a message to your test results:

1 In the test tree, select a step and choose **Insert** > **Step** > **Report**, or right-click a step and choose **Insert Report Step**. The Insert Report dialog box opens.



2 Select the status that will result from this step from the **Status** list.

Status	Description
Passed	Causes the status of this step to be passed and sends the specified message to the report.
Failed	Causes the status of this step to be failed and sends the specified message to the report. When this step runs, the test fails.
General	Sends a message to the report without affecting the pass/fail status of the test.
Warning	Sends a warning message to the report, but does not cause the test to stop running, and does not affect the pass/fail status of the test.

3 In the **Name** box, type a name for the test step. For example, “Password edit box”.

4 In the **Details** box, type a detailed description of this step to insert in your test results. For example, "Password edit box does not exist".

5 Click **OK**. A report step is inserted into the test tree and a **ReportEvent** statement is inserted into your script in the expert view. For example:

```
Reporter.ReportEvent 1, "Password edit box", "Password edit box does
    not exist"
```

Where "ReportEvent 1" indicates the status of the report (failed), "Password edit box" is the report name, and "Password edit box does not exist" is the report message.

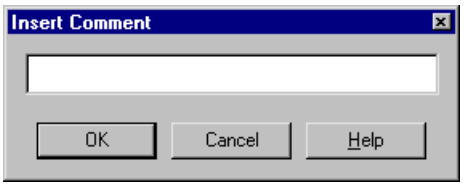
For more information on test results, see Chapter 18, "Analyzing Test Results"

Adding Comments

While programming, you can add comments to your tests. A *comment* is an explanatory remark in a program. When you run a test, QuickTest does not process comments. Use comments to explain sections of a test in order to improve readability and to make tests easier to update.

To add a comment:

1 In the test tree, right-click a step and choose **Insert** > **Step** > **Comment**. The Insert Comment dialog box opens.



2 Type a comment and click **OK**.

A comment statement is added to your test. If you are working in Tree View, the icon indicates a comment. In the Expert View, a comment is specified with apostrophes (').

22

Testing in the Expert View

In QuickTest, test scripts are composed of statements coded in Microsoft's programming language, VBScript. This chapter provides a brief introduction to VBScript and shows you how to enhance your test scripts using a few simple programming techniques.

This chapter describes:

- Understanding the Expert View
- Programming in the Expert View
- Enhancing Tests with Comments, Calculations, and Control-Flow Statements
- Accessing Runtime Object Properties and Methods

About Testing in the Expert View

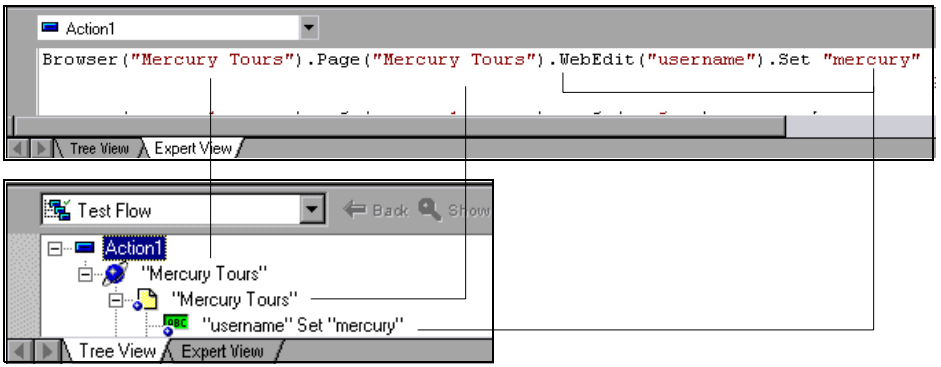
The Expert View provides an alternative to the Tree View for testers who are familiar with VBScript. In the Expert View, you can view the recorded test in VBScript and enhance it with programming.

In the Expert View you can also add methods manually instead of from the Method wizard. For information on using the Method wizard, see Chapter 21, "Enhancing Your Tests with Programming."

Understanding the Expert View

QuickTest can display tests you record in two formats:

- the Tree View, where QuickTest displays the object hierarchy in an icon-based tree
- the Expert View, where QuickTest displays the object hierarchy in VBScript



Note that in the diagram above, the object hierarchy is identical in both views.

Each line of VBScript in the Expert View represents a step in the test. The example above represents a step in the test in which the user inserts the name "mercury" into an edit field. The hierarchy of the step enables you to see the name of the site, the name of the page, the name of the object in the page, and the name of the method performed on the object. When you record your test, QuickTest records the operations you perform on your application in terms of the objects in it. It identifies the objects in an application by specific names (e.g. a button or a list) and the method performed on the object (e.g. click or select).

To further understand how a step in the Expert View corresponds with a step in the Tree View, examine the table below:

Tree View	Expert View	Description
 "Mercury Tours"	Browser (“Mercury Tours”)	The name of the Web site is “Mercury Tours”.
 "Mercury Tours"	Page (“Mercury Tours”)	The name of the current page in the Web site is “Mercury Tours”.
 "username"	WebEdit(“username”)	The name of the edit field upon which the action is performed is “username”.
Set "mercury"	Set “mercury”	The name of the method performed on the edit box is “Set”. The name inserted into the edit box is “mercury”.

An object’s logical name is displayed in parentheses following the object type. In the following example, the object type is Browser, and the logical name of the Browser is “Mercury Tours”:

```
Browser ("Mercury Tours")
```

The object types in the object hierarchy are separated by a period. In the following example, Browser and Page are two separate objects:

```
Browser("Mercury Tours").Page("Mercury Tours")
```

The method performed on the object always is displayed at the end of the line of script. In the following example, the word “mercury” is inserted in the “username” edit box using the **Set** method:

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set
    "mercury"
```

For a complete list of objects and their associated methods, choose **Help** > **QuickTest Object Model Reference** to open the *QuickTest Object Model Reference*.

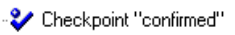
Checkpoints

In QuickTest, you create checkpoints on pages, text strings, objects, and tables. When you create a checkpoint in the Tree View, QuickTest creates a corresponding line in VBScript in the Expert View. It uses the **Check** method to perform the checkpoint.

For example, in the following statement QuickTest performs a check on the word "confirmed":

```
Browser("Mercury Tours").Page("Flight Confirmation").
    Check Checkpoint("confirmed")
```

The corresponding step in the Tree View is displayed as follows:



Note: You cannot insert checkpoints into your test while in the Expert View when you are not recording. Use the Tree View to insert and modify checkpoints. For more information on inserting and modifying checkpoints, see Chapter 6, "Creating Checkpoints."

Parameters

You can use QuickTest to enhance your tests by parameterizing values in the test. A *parameter* is a variable that is assigned a value from outside the test in which it is defined. When you create a parameter in the Tree View, QuickTest creates a corresponding line in VBScript in the Expert View.

QuickTest calls the values of a parameterized object from the data table using the following syntax:

method_name **DataTable (** *parameterID*, *sheetID* **)**

method_name	The name of the method that QuickTest executes on the parameterized object.
DataTable	The data table object.
parameterID	The name of the column in the data table.
sheetID	The name of the sheet. If the parameter is a global parameter, "dtGlobalSheet" is displayed.

Note: You cannot create parameters from the Expert View. Use the Tree View to create parameters. For more information on parameterization, see Chapter 10, "Parameterizing Tests."

For example, suppose you are creating a test on the Mercury Tours site, and you select "Paris" as your destination. The following statement is inserted into your test in the Expert View:

```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select "Paris"
```

Now suppose you want to parameterize the destination, and you create a "Departure" column in the data table. The previous statement is modified to the following:

```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select
    DataTable("Departure",dtGlobalSheet)
```

where **Select** is the method name, DataTable is the object, Departure is the name of the column in the data table, and dtGlobalSheet is the name of the sheet in the data table.

Programming in the Expert View

The Expert View displays in VBScript the steps you executed while recording your test. After you record your test, you can increase its power and flexibility by adding recordable and non-recordable VBScript statements. You can add statements that perform operations on objects or retrieve information from your site. For example, you can add a step that checks that an object exists, or you can retrieve the return value of a method.

The objects in QuickTest are divided by environment.

QuickTest environments include Web objects, ActiveX objects, DataTable objects, Utility objects, Java objects, Macromedia Flash objects, and Real Player objects, and standard Windows objects.

Note: Java objects, Macromedia Flash objects, and Real Player objects are supported in QuickTest Professional only.

Most objects have corresponding methods. For example, the **Back** method is associated with the Browser object.

In the following example, the user inserts "mercury" in the User Name edit box while recording. The following line is recorded in the Expert View:

```
Browser ("Mercury_Tours"). Page ("Mercury_Tours"). WebEdit ("username").
    Set"mercury"
```

Browser object	Mercury_Tours
Page object	Mercury_Tours
WebEdit object (edit box)	username

The **Set** method sets the "mercury" text into the WebEdit object.

In the following example, the user selects “Paris” from the Departure City drop-down list while recording. The following line is recorded in the Expert View:

```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select "Paris"
```

Browser object	Mercury Tours
Page object	Find Flights
WebList object (combo box or list box)	depart

The **Select** method selects Paris in the WebList object.

Note: For more information on QuickTest objects and methods, refer to the *QuickTest Object Model Reference*. To open this, choose **Help** > **QuickTest Object Model Reference**.

Generating a Method for an Object

In the Expert View you can generate methods. You can also generate methods in the Tree View using the Method wizard. For additional information, see Chapter 21, “Enhancing Your Tests with Programming.”

By default, QuickTest displays the syntax for methods as you type. You can disable or enable this **Statement Completion** option in the Editor Options dialog box. For additional information, see Chapter 26, “Customizing the Expert View.”

To generate a method in the Expert View:

1 In the Expert View, type a period after the object upon which you want to perform the method.

2 A list of the available methods for the object is displayed.

Double-click a method in the list. QuickTest inserts the method into the line of script.

3 If the method contains arguments, and the Statement Completion option is enabled, QuickTest displays the syntax of the method.

In the above example, the **Set** method has one argument, called *Text*. The argument name represents the text to enter in the edit box.

4 Insert an argument after the method.

Tip: When programming in VBScript, use parentheses around method parameters if the method is expected to return a value. If the method does not return a value, parentheses are optional.

Note: To find the syntax for a method, refer to the *QuickTest Object Model Reference*.

Using Descriptive Programming

When you record an operation on an object, QuickTest adds the appropriate test object to the Object Repository. Once 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 enter the logical name 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 logical name of an edit field. The edit field is located on a page with the logical name "Mercury Tours" and the page was recorded in a browser with the logical name "Mercury Tours".

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username")
```

Because each object in the Object Repository has a unique logical name, this is all you need to describe the object. During the test run, QuickTest finds the object in the Object Repository and uses the other property values stored for that test object to identify the object in your Web site or application.

You can also add statements to perform methods on objects without using the Object Repository. To do this, you need to enter more information in the description of the object in order to uniquely describe the object so that QuickTest can identify the object during the test run.

For example, suppose you recorded on a Web form in your Web site. Then, after you created your test, an additional edit field was added to the form.

The Browser and Page objects already exist, but the new edit object does not. Rather than recording a new step in your existing test, you can add a statement to the script that describes the new edit object, and performs a **Set** method on it.

You describe the object by using as many *property:=value* pairs as necessary to uniquely identify the object.

The general syntax is:

*TOClass***("***PropertyName1***:=***ProperyValue1***",** *...* **,**
"*PropertyNameX***=***ProperyValueX***")**

TOClass: The test object class.

PropertyName:=PropertyValue: The test object property and its value. Each property:=value pair should be separated by commas.

When you use descriptive programming, QuickTest creates a live instance of the test object and assigns it the property values from your description. QuickTest uses the property values in the live instance of the test object to identify the object during the test run.

The statement below creates a WebEdit test object in the Mercury Tours page with the Name author and an index of 3. When the test runs, 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"
```

For more information about working with test objects, see Chapter 4, "Managing Test Objects.".

If you want to use the same object several times in one test, you can assign the object you create to a variable.

For example, instead of entering:

```
Window("Title:=Notepad").Move 50, 50
Window("Title:=Notepad").WinEdit("AttachedText:=Find what:").Set "hello"
Window("Title:=Notepad").WinButton("Caption:=Find next").Click
```

You can enter:

```
Set MyWin = Window("Title:=Notepad")
MyWin.Move 50, 50
MyWin.WinEdit("AttachedText:=Find what:").Set "hello"
MyWin.WinButton("Caption:=Find next").Click
```

Alternatively, you can use a With statement:

```
With Window("Title:=Notepad")
    .Move 50, 50
    .WinEdit("AttachedText:=Find what:").Set "hello"
    .WinButton("Caption:=Find next").Click
End With
```

For more information about the **With** statement, see ""With"Statement," on page 372.

Using Descriptive Programming for the WebElement Object

The *WebElement* object is 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 descriptive programming 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
```

QuickTest clicks on the first Web object in the Mercury Tours page with the name UserName.

For more information about the WebElement object, refer to the *QuickTest Object Model Reference*.

Using the Index Property in Descriptive Programming

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.

Note that *Index* property values are object-specific. Thus, if you use "Index:=3" to describe a WebEdit test object, QuickTest searches for the fourth WebEdit object in the page.

If you use "Index:=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")
```

Enhancing Tests with Comments, Calculations, and Control-Flow Statements

QuickTest enables you to incorporate decision-making into your test by adding conditional statements that control the logical flow of your test. In addition, you can define messages in your test that QuickTest sends to your test results. To improve the readability of your tests, you can also add comments to your test.

For information on how to use these programming concepts in the Tree View, see Chapter 21, “Enhancing Your Tests with Programming.”

Comments

A comment is a line or part of a line in a test script that is preceded by an apostrophe (‘). When you run a test, QuickTest does not process comments. Use comments to explain sections of a test script in order to improve readability and to make tests easier to update. Comments are displayed in green. For example:

```
‘Sets the word “mercury” into the “password” edit field.
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").
    Set "mercury"
```

Note: You can also add a comment line using VBScript’s **Rem** method. For additional information, refer to the *VBScript Reference* (choose **Help > VBScript Reference**).

Performing Calculations

You can create tests 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:

+	addition
-	subtraction
-	negation (a negative number - unary operator)
*	multiplication
/	division
^	exponent

In the following example, the multiplication operator is used to calculate the total 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 execute one or more statements a specified number of times. It has the following syntax:

for *counter* = *start* **to** *end* [**Step** *step*]
    statement
Next

counter	The variable used as a counter.
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. The default is 1.
statement	The statement to be executed during the loop.

In the following example, QuickTest calculates the factorial value of the number of passengers using the For statement.

```
number = Browser("Mercury Tours").Page("Find
Flights").WebEdit("numPassengers").GetROProperty("value")
total = 1
For i=1 to number
    total = total * i
next
MsgBox "!" & number & "=" & total
```

"For...Each" Statement

A **For...Each** loop instructs QuickTest to execute 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	a variable representing the element in the array
array	the name of the array
statement	a statement or series of statements to be executed during the loop

"Do...Loop" Statement

The **Do...Loop** statement instructs QuickTest to execute 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

condition	a condition to be fulfilled
statement	a statement or series of statements to be executed during the loop

QuickTest calculates the factorial value of the number of passengers using the Do...Loop.

```
number = Browser("Mercury Tours").Page("Find
Flights").WebEdit("numPassengers").GetROProperty("value")
total = 1
i = 1
do while i <= number
    total = total * i
    i = i + 1
Loop
MsgBox "!" & number & "=" & total
```

"While" Statement

A **While** statement instructs QuickTest to execute a statement or series of statements while a condition is true. It has the following syntax:

While *condition*
    statement
Wend

In the following example, QuickTest performs a loop using the **While** statement while the number of passengers is fewer than four. Within each loop, QuickTest increments the number by one.

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
while passengers < 4
    passengers = passengers + 1
wend
```

"If...Then...Else" Statement

The **If...Then...Else** statement instructs QuickTest to execute a statement or a series of statements based on specified conditions. If a condition is not fulfilled, the next **elseif** or **else** statement is examined. It has the following syntax:

If *condition* **Then**
    statement
ElseIf
    statement
Else
    statement
EndIf

condition	condition to be fulfilled
statement	statement to be executed

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
```

"Dim" Statement

The **Dim** statement is used to declared variables of all types, including strings, integers, and arrays. Use the **Dim** statement at the beginning of the procedure. It has the following syntax:

Dim *variable* **[(***subscript***)]**

variable	the name of the variable
subscript	the dimensions of the array

In the following example, the Dim statement is used to declare the "passengers" variable.

```
Dim passengers
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
```

"With"Statement

The **With** statement enables you to omit the object from a block of lines (from the **With** statement until the **End With** statement), so that lines beginning with "**.**" are treated as though the object is written before the "**.**" It has the following syntax:

With *object*
    statements
End With

object	an object or a function that returns an object
statements	one or more statements to be executed on an object

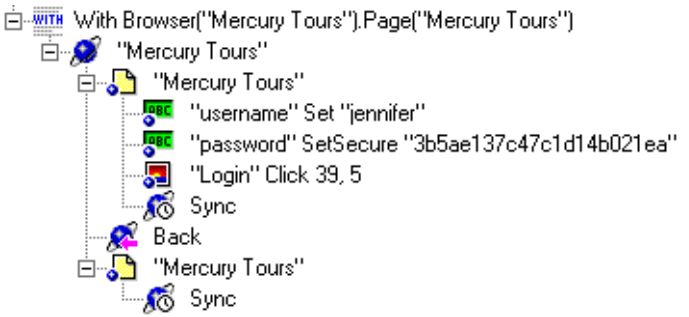
The **With** statement is a "shortcut" in VBScript. For example, you can write:

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username")
    .Set "jennifer"
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("password")
    .SetSecure "3b5ae137c47c1d14b021ea"
Browser("Mercury Tours").Page("Mercury Tours").Image("Login").Click 39,5
Browser("Mercury Tours").Page("Welcome to Mercury").Sync
Browser("Mercury Tours").Back
Browser("Mercury Tours").Page("Mercury Tours").Sync
```

more concisely as:

```
With Browser("Mercury Tours").Page("Mercury Tours")
.WebEdit("username").Set "jennifer"
.WebEdit("password").SetSecure "3b5ae137c47c1d14b021ea"
.Image("Login").Click 39,5
.Sync
Browser("Mercury Tours").Back
.Sync
End With
```

This is displayed as follows in the Tree View:



Note that each line following the **With** statement is displayed as if it were written out fully in the Expert View, with the object name at the beginning of each line.

Note: QuickTest includes Microsoft's *VBScript Language Reference*. The VBScript Language reference describes VBScript in detail. To open this reference, choose **Help** > **VBScript Reference**.

Accessing Runtime Object Properties and Methods

When you work in the Expert View, you can access actual properties and methods of objects in your Web site or application that you may not be able to access using QuickTest's test object properties and methods.

Tip: If you do not know the properties and methods of objects in your Web site or application, you can view them using the Object Spy. For information on the Object Spy, see Chapter 3, "Understanding the Test Object Model."

Retrieving Runtime Object Properties

You can use the **GetROProperty** method to retrieve runtime object properties. This is useful to activate these properties using a specific method, (e.g. using the **Click** method).

The **GetROProperty** method has the following syntax:

RunTimeObjectName**.GetROProperty (**Property, in_PropData**)**

For additional information about the **Click** and **GetROProperty** methods, refer to the *QuickTest Object Model Reference*.

Activating Runtime Object Methods

In the Expert View, you can use the "Object" property to activate the internal methods of any runtime object. It has the following syntax:

```
RunTimeObjectName(RunTimeObjectName).Object.Method_to_activate( )
```

For additional information about the "Object" property, refer to the *QuickTest Object Model Reference*.

23

Working with QuickTest—for Power Users

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 Tests
- Working with Dynamic Web Content
- Advanced Web Issues
- Test Maintenance
- Testing Localized Applications

Recording and Running Tests

- **How does QuickTest capture user processes?**

  QuickTest hooks the browser (Netscape, Microsoft Internet Explorer, or AOL). As the user navigates the Web-based application, QuickTest records the user actions. (For information about modifying which user actions are recorded, see Chapter 20, "Configuring Event Recording.") QuickTest can then run the test by running the steps as they originally occurred.

- **How does QuickTest record and identify objects on Web pages?**

  QuickTest can record all Web objects on a Web page. Each HTML tag is considered a Web object. QuickTest identifies each object by its HTML tag and logical name and stores the descriptions of each object in the memory.

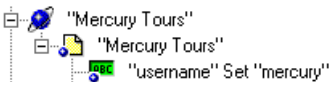
➤ **How can I record on objects or environments not supported by QuickTest?**

There are two ways to record on such objects. You can either define *virtual objects* for objects that behave like standard objects and then record in the standard recording mode, or you can record your clicks and key-board input based on coordinates in the *low-level recording* mode.

For more information on defining virtual objects, see Chapter 13, "Learning Virtual Objects." To record in low-level recording mode, follow the instructions below:

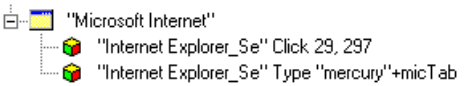
While recording, select **Test** > **Low Level Recording** in QuickTest. The record mode changes to low-level recording, and all of your clicks are recorded based on mouse coordinates. When QuickTest runs the test, the cursor retraces the recorded clicks.

For example, if you type the word mercury into a user name edit box and then click the Tab key while in standard recording mode, your test tree and script would be displayed as follows:




```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set
"mercury"
```

If you perform the same action while in low-level recording mode, QuickTest would record the click in the user name box, followed by the keyboard input including the tab key. The test tree and script would look like this:




```
Window("Microsoft Internet").WinObject("Internet Explorer_Se").Click 29,297
Window("Microsoft Internet").WinObject("Internet Explorer_Se").Type "mercury"
    + micTab
```

Working with Dynamic Web Content

➤ **How can I record and run tests on objects that change dynamically from viewing to viewing?**

Sometimes the content of objects in a Web page 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 4, "Managing Test Objects."

➤ **How can I check that a spawned window exists (or does not exist)?**

Sometimes a link in one window spawns another window. Use the **Exist** method to check whether or not a spawned window exists. For example:

```
Browser("Window_logical_name").Exist
```

For additional information about the **Exist** method, refer to the *QuickTest Object Model Reference*.

➤ **How does QuickTest record on dynamically generated URLs and Web pages?**

QuickTest actually clicks on 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.

➤ **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).

➤ **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.

Test Maintenance

➤ **How do I maintain my test when my application changes?**

The way to maintain a test when your application changes depends on how much your application changes. This is one of the main reasons you should create a small group of tests rather than one large test for your entire application. When your application changes, you can rerecord part of a test. If the change is not significant, you can manually edit a test to update it.

You can also use QuickTest's action feature to design more modular and efficient tests. While recording, you divide your test into several actions, based on functionality. When your application changes, you can rerecord an action, without changing the rest of the test. For additional information, see Chapter 14, "Working with Actions."

Testing Localized Applications

➤ **I am testing localized versions of a single application, each with localized user interface strings. How do I create efficient tests in QuickTest?**

You can parameterize these user interface strings using parameters from the global Environment variable list. This is a list of variables and corresponding values that can be accessed from any test. For additional information, see Chapter 10, "Parameterizing Tests."

➤ **I am testing localized versions of a single application. How can I efficiently input different data in my tests, depending on the language of the application?**

You can create an external data table for each localized version of your application. When you change the localized version of the application you are testing, you simply switch the data table for your test. For additional information, see Chapter 15, "Working with Data Tables."

➤ **I am testing localized versions of a single application. How can I efficiently instruct QuickTest to handle exceptions, when the exception strings vary, depending on the language of the application?**

QuickTest stores information about handling unexpected events and errors in an external exception configuration file. This file, *Exception.inf*, is located in the *[QuickTest installation]\dat* folder. When you change the localized version of the application you are testing, you can simply modify the *Exception.inf* exception configuration file by adding entries that are relevant to the new version. For additional information, see Chapter 16, "Handling Unexpected Events and Errors."

Part VI

Configuring QuickTest

24

Setting Global Testing Options

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

This chapter describes:

- Setting Global Testing Options
- Selecting Global Testing Options

About Setting Global Options

Global testing options affect how you record and run tests, as well as the general appearance of QuickTest. For example, you can choose not to display the Welcome screen when QuickTest starts, or set the timing-related settings used by QuickTest when running a test. The values you set remain in effect for all tests and for subsequent testing sessions.

You can also set testing options that effect only the test currently open in QuickTest. For more information, see Chapter 25, “Setting Testing Options for a Single Test.”

Setting Global Testing Options

Before you record or run a test, you can use the Options dialog box to modify your testing options. The values you set remain in effect for all tests.

To set global testing options:

1 Choose **Tools** > **Options**.

The Options dialog box opens. It is divided by subject into five tabbed pages.

2 To choose a page, click a tab.

3 Set an option, as described in the following section "Selecting Global Testing Options," on page 386.

4 To apply your changes and keep the Options dialog box open, click **Apply**.

5 When you are done, click **OK** to save your changes and close the dialog box.

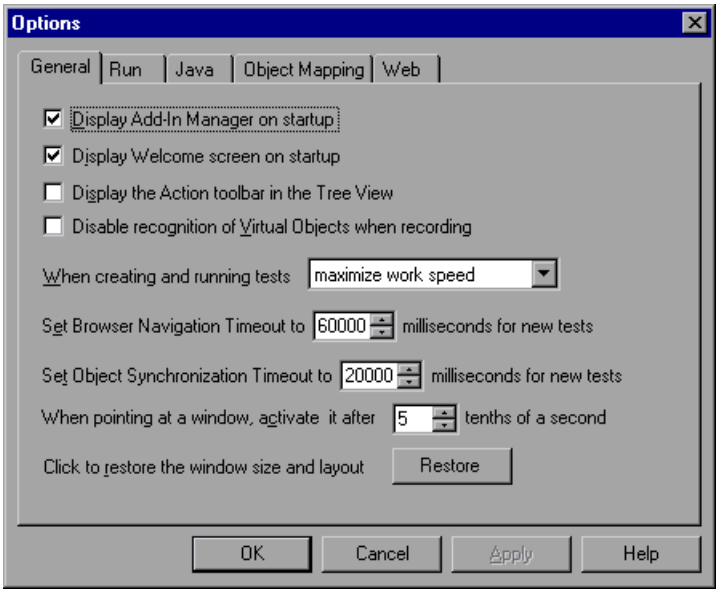
Selecting Global Testing Options

The Options dialog box contains the following tabbed pages:

Tab Heading	Subject
General	Options for global settings.
Run	Options for running tests.
Java	Options for setting how QuickTest records and runs tests on a Java applet or application, or the objects within them. **Note:** Testing on Java objects is supported only in QuickTest Professional.
Object Mapping	Options for mapping an object, whose class cannot be determined, to a standard class.
Web	Options for configuring recording and test run performance in the Web environment.

General Testing Options

The General tab options affect the general appearance of QuickTest and the default run time settings for all new tests.



The General tab includes the following options:

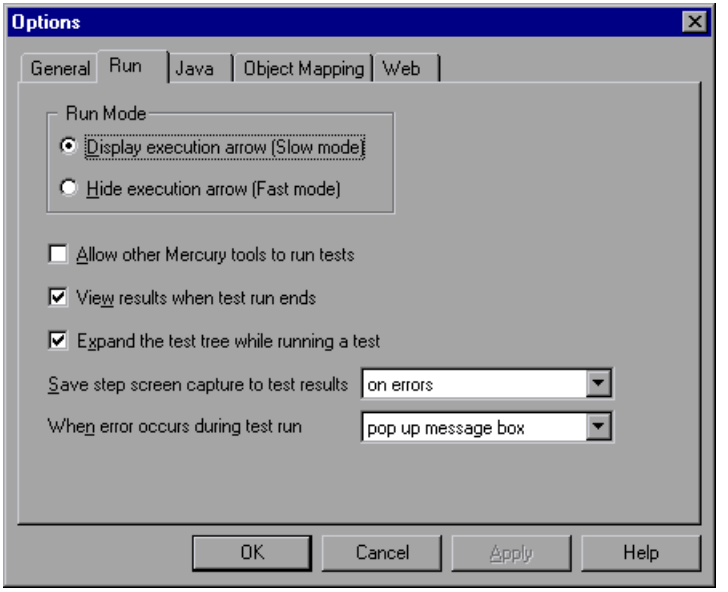
Option	Description
Display Add-In Manager on startup	Determines whether the Add-In Manager is displayed when starting QuickTest. For information on working with the Add-In Manager, see “Loading QuickTest Add-Ins,” on page 23.
Display Welcome screen on startup	Determines whether the Welcome screen is displayed when starting QuickTest.
Display the Action toolbar in the Tree View	Determines whether the Action toolbar is displayed in the Tree View. If your test contains reusable or external actions, this option is automatically selected and disabled.

Option	Description
Disable recognition of Virtual Objects when recording	Determines whether the defined virtual objects stored in the Virtual Object Manager are recognized when recording.
When creating and running tests	**When creating and running tests:** Sets the optimization to either maximize the speed of creating and running tests, or to minimize the disk space used by the test and test results. • **maximize work speed:** Maximizes the speed at which tests are created and run. • **minimize used disk space:** Minimizes the disk space used by the test and test results. **Note:** The **minimize used disk space** option may affect how quickly tests are created and run in certain environments. For best results, try changing the mode to see whether or not it affects the speed at which tests are created and run in the environments you are testing.
Set Browser Navigation Timeout	Sets the navigation interval (in milliseconds). This is the default setting of the Browser Navigation Timeout interval for all new tests. To change this setting locally for your test, use the Browser Navigation Timeout option in the Web tab of the Test Settings dialog box. For more information, see "Web Test Settings," on page 412.
Set Object Synchronization Timeout	Sets the default object synchronization interval (in milliseconds). This is the default setting of the Object Synchronization Timeout interval for all new tests. To change this setting locally for your test, use the Object Synchronization Timeout option in the Run tab of the Test Settings dialog box. For more information, see "Run Test Settings," on page 409.

Option	Description
Activate window after	Specifies the time (in tenths of a second) that QuickTest waits before it sets focus on the window under the cursor.
Restore	Restores the QuickTest window so that it displays the default panes in the default sizes.

Test Run Options

The Run tab options affect how QuickTest runs tests and displays test results.

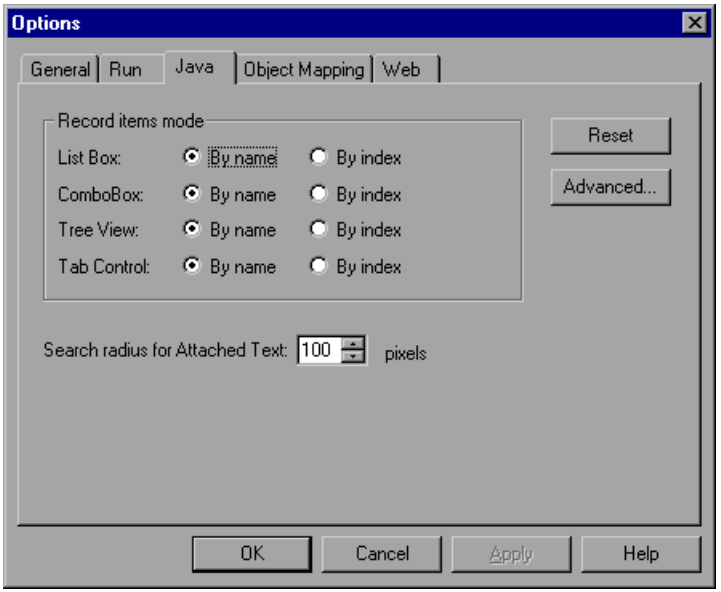


The Run tab includes the following options:

Option	Description
Run mode - Display execution arrow (Slow mode)	Instructs QuickTest to run your test with the execution arrow in the left margin of the test, marking each step or statement as it is interpreted. **Note:** You must have Microsoft Script Debugger installed in order to enable this mode. For more information, refer to the *QuickTest Installation Guide*.
Run mode - Hide execution arrow (Fast mode)	Instructs QuickTest to run your test without the execution arrow in the left margin of the test.
Allow other Mercury tools to run tests	Enables TestDirector, WinRunner, and Test Batch Runner to run tests.
View results when test run ends	Instructs QuickTest to display the test results automatically following the test run.
Expand the test tree while running a test	Instructs QuickTest to expand the Tree View during the test run.
Save step screen capture to test results	Instructs QuickTest when to save screen captures of the application to the test results. Choose an option from the list: • **on errors** saves screen captures in steps that fail. • **always** saves screen captures whether the test run fails or passes. • **on errors and warnings** saves screen captures in steps that fail or generate a warning.
When error occurs during test run	Specifies how QuickTest responds to an error during a test run. Choose an option from the list: • **pop up message box** displays an error message dialog box when an error occurs. • **proceed to next iteration** jumps to the next iteration when an error occurs. • **stop run** stops the test when an error occurs.

Java Testing Options

The Java tab option enables you to configure how QuickTest learns the descriptions of Java objects, records tests, and runs tests on Java applets or applications.



Note: The Java tab is only available when the Java support is installed. Java support is only available in QuickTest Professional.

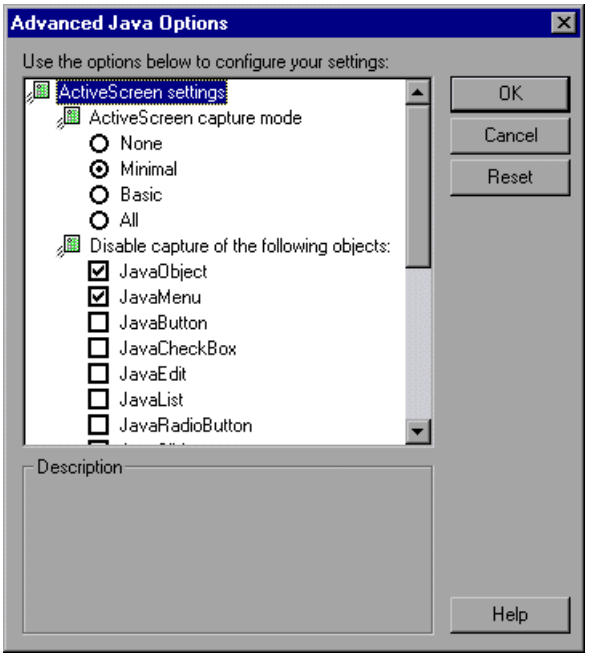
The Java tab includes the following options:

Option	Description
Record items mode	Determines how operations on items within List Box, ComboBox, TreeView, and Tab objects are recorded. Select one of the following options for each object: • **By Name** records operations on an item within the object (i.e. selected list item, tab, etc.) according to the item's name. • **By Index** records operations on an item within the object (i.e. selected list item, tab, etc.) according to the item's position within the Java object. **Default = By Name**
Search radius for Attached Text	Sets the maximum distance in pixels to search for attached text. **Default value: 100**
Reset	Resets the value of the Java test settings to their default values.
Advanced	Opens the Advanced Java Options dialog box.

Note: Additional options can be set and retrieved from the script using the **SetAUTVar** and **GetAUTVar** methods. For more information see Chapter 31, "Testing Java Applets and Applications," or refer to the *QuickTest Object Model Reference*.

Advanced Java Options

The Advanced Java Options dialog box enables you to set ActiveScreen and property capture, event, and record mode preferences.



The Advanced Java options include the following **ActiveScreen settings**:

Option	Description
ActiveScreen capture mode	Sets the quantity of properties captured for each object in the applet or application when it is captured for the ActiveScreen. Choose one of the following modes: • **None:** no ActiveScreen capture. • **Minimal**: captures all properties and values in the first step in which the applet or application test object is used. In the succeeding steps of the same applet or application, only the properties and values of the object specified in the step are captured. • **Basic**: captures basic properties and the corresponding values for all objects in all steps. This option is useful for heavy applets and applications with many objects. • **All**: captures all properties and values of the objects in the applet or application for all steps. **Default value: "Minimal"**
Disable capture of the following objects	Prevents QuickTest from capturing the objects of the selected test object types in the ActiveScreen. The object will be visible in the ActiveScreen as an image only. **Default = JavaObject and JavaMenu are selected.**

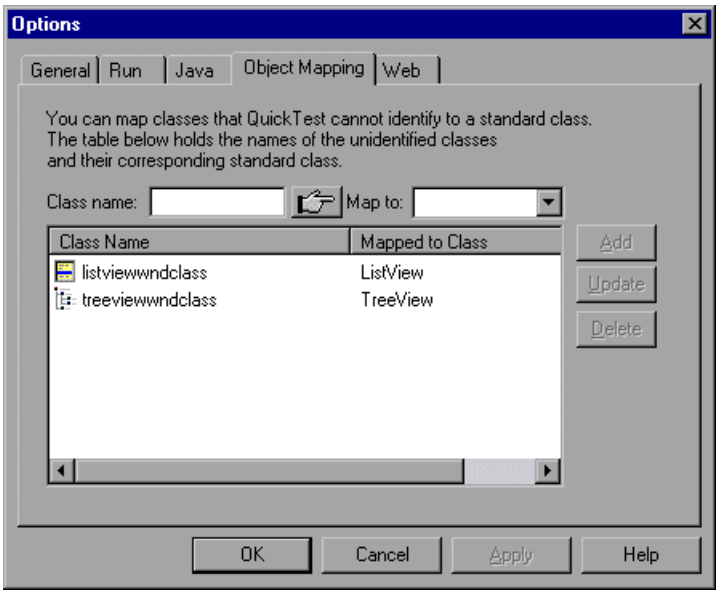
The Advanced Java options include the following **Record and Run settings**:

Option	Description
AWT event model	Sets the event model used to send events to the objects in the applet or application. Choose one of the following models: • **New - JDK 1.1 and Up**: applies the new event model. • **Old - JDK 1.0:** applies the old event model. • **Auto-Detect:** enables QuickTest to use the old event model for AWT objects and to use the new event model for all other toolkit objects. **Default value:** "**Auto-Detect**"
JavaTable record mode	Sets the record mode for a table object. Choose one of the following modes: • **Context Sensitive**: records in Context Sensitive mode. • **Analog**: records only low-level (Analog) table methods: **ClickCell**, **DoubleClickCell**, and **Drag**. **Default value: "Context Sensitive"**

Object Mapping

The Object Mapping tab enables you to map an object of an unidentified class to a standard class.For example, if your site has a button that cannot be identified, this button is recorded as a Web object. You can teach QuickTest to identify the object using the standard button class. Then, when you click the button, the operation is recorded as an **object.Click x, y** statement in the test script.

Note that an object that cannot be identified should be mapped only to a standard class with comparable behavior. For example, you cannot map an object that behaves like a button to the edit class.



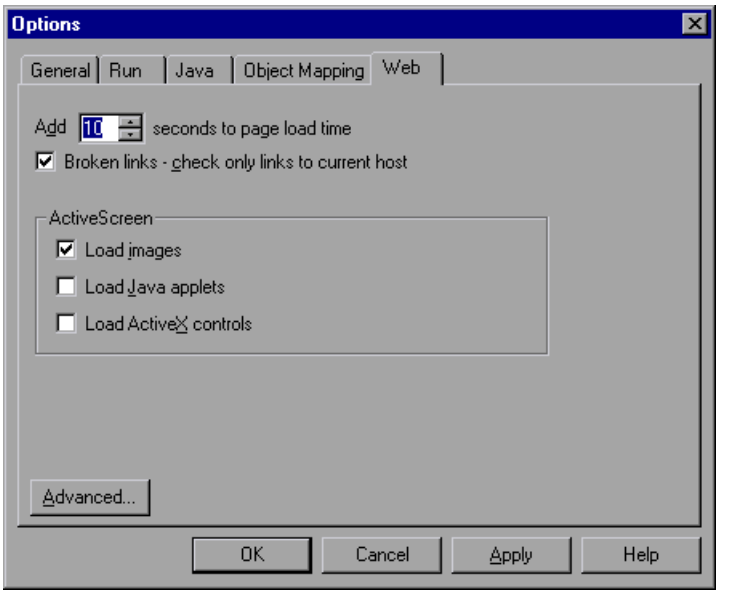
The Object Mapping tab includes the following options:

Option	Description
Class Name	Adds a new class name. Use the pointing hand to click the object whose class you want to add. A name is displayed in the **Class name** box. Click **Add** to add it to the **Class Name** list. If you want to map it to a standard class, choose one from the **Map to** list box, and click **Add**.
Mapped to Class	Lists all standard classes. To map an object that cannot be identified to a standard class, select the object in the **Class Name** list and select a standard class name from the **Map to** list. Click **Update** to change the class in the **Mapped to Class** list for the object.
Add	Adds the new class indicated in the **Class name** box to the **Class Name** list.

Option	Description
Update	Updates the selected class by mapping it to the standard class indicated in the **Map to** list.
Delete	Deletes the selected class from the map list.

Web Options

The Web tab options determine QuickTest's behavior on a Web page when recording and running tests.

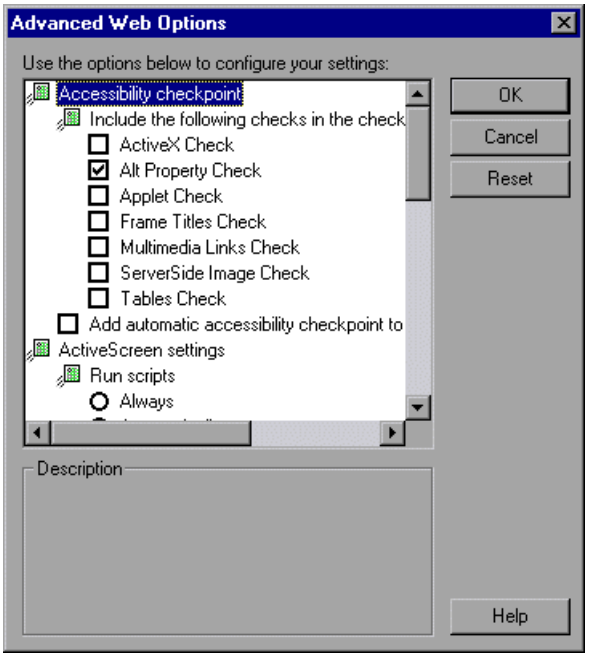


The Web tab includes the following options:

Option	Description
Add seconds to page load time	Instructs QuickTest to add a specified number of seconds to the load time of the page. **Note:** This option is a safeguard that prevents the test from failing in the event that the amount of time it takes for a page to load during the test 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.
Load images	Instructs QuickTest to load images from your browser page to the ActiveScreen pane.
Load Java applets	Instructs QuickTest to load Java applets from your browser page to the ActiveScreen pane. If this option is cleared, a default Java image is displayed in the Active Screen for all Java applet objects.
Load ActiveX controls	Instructs QuickTest to load ActiveX controls from your browser page to the ActiveScreen pane. If this option is cleared, a default ActiveX image is displayed in the ActiveScreen for all ActiveX control objects.
Advanced	Opens the Advanced Web Options dialog box, where you can modify record and run settings.

Advanced Web Options

The Advanced Web Options dialog box enables you to modify how QuickTest records and runs tests on Web pages.



You can add an accessibility checkpoint to check that the Web page conforms to the W3C Web Content Accessibility Guidelines. The Advanced Web Options dialog box includes the following **Accessibility checkpoint** options:

Option	Description
Include the following checks in the checkpoint	Instructs QuickTest to check the following selected accessibility elements: • **ActiveX Check** retrieves a list of all ActiveX and Multimedia objects. • **Alt Property Check** checks that the <alt>/<longdesc> tag exists for all relevant objects (such as images). • **Applet Check** retrieves a list of Java applets. • **Frame Titles Check** checks that all frames have titles. • **Multimedia Links Check** retrieves a list of links to multimedia objects. • **ServerSide Image Check** retrieves a list of Server-side images. • **Tables Check** retrieves a list of tables.
Add automatic accessibility checkpoint to each Web page while recording	Instructs QuickTest to automatically add an accessibility checkpoint to each Web page while recording.

You can specify how QuickTest displays Web pages in the ActiveScreen. The Advanced Web Options dialog box includes the following **ActiveScreen settings**:

Option	Description
Run scripts	Specifies whether QuickTest runs scripts when displaying Web pages in the ActiveScreen, according to one of the following options: • **Always** runs scripts whenever loading a page in the ActiveScreen. • **Automatically** runs scripts as needed according to the page that is displayed. • **Never** prevents scripts from running when loading a page in the ActiveScreen.

You can check that expected and actual page properties are identical. The Advanced Web Options dialog box includes the following **Automatic Web page checkpoint** options:

Option	Description
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. The automatic page checkpoint includes the checks that you select from among the following options: **Note:** If you are testing a Web page with dynamic content, using automatic page checkpoints may cause the checkpoint to fail as these checkpoints assume that the page content will be identical when the test is recorded and when it is run. • **Broken links** displays the number of broken links contained in the page during the test run. • **HTML source** checks that the expected source code is identical to the source code during the test run.

Option	Description
	• **HTML tags** checks that the expected HTML tags in the source code are identical to those in the test run. • **Image source** checks that the expected source paths of the images are identical to the sources in the test run. • **Links URL** checks that the expected URL addresses for the links are identical to the addresses in the source code during the test run. • **Load time** checks that the expected amount of time it takes for the page to load during the test run is less than or equal to the amount of time it took during the record session. • **Number of images** checks that the expected number of images is identical to the number displayed in the test run. • **Number of links** checks that the expected number of links is identical to the number displayed in the test run.
Ignore automatic checkpoints while running tests	Instructs QuickTest to ignore the automatically added page checkpoints while running your test.

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

Option	Description
Disable ActiveScreen capture	Disables the screen capture of all steps in the ActiveScreen.
Force static source	Prevents QuickTest from saving dynamic changes to the HTML source.

Option	Description
Record coordinates	Records the actual coordinates relative to the object for each operation.
Record Navigate for all navigation operations	Records a Navigate statement each time a Frame URL changes.

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

Option	Description
Browser cleanup	Closes all open browsers when the test or iteration is finished.
Run only click	Runs click events using the mousedown, mouseup and the click or using only the click.
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.
Use optional algorithm	Instructs QuickTest to use optional properties when an object cannot be identified using only obligatory properties. *Obligatory properties* are properties that QuickTest always uses to identify a specific type of object. For example, QuickTest uses the HTML tag and alt obligatory properties to identify objects of the image class. These properties are listed in the Object Repository for this class of object, where they can be modified. *Optional properties* are test object properties that cannot be modified. For example, QuickTest uses the class and html id optional properties to identify objects of the image class.

25

Setting Testing Options for a Single Test

You can control how QuickTest records and runs specific tests by setting testing options.

This chapter describes:

- Setting Testing Options for a Single Test
- Selecting Testing Options for a Single Test
- Setting Navigation Fallback Properties

About Setting Testing Options for a Single Test

You can set testing options that affect how you record and run a specific test. For example, you can instruct QuickTest to run a parameterized action for only certain lines in the table in the Data pane. You can also teach QuickTest to recognize a specific object in your test as a standard object. These testing options are saved when you save the test.

You can also set testing options for part of the test from within a test. For more information, see Chapter 27, “Setting Testing Options from a Test Script.”

You can also set testing options that affect all tests. For more information, see Chapter 24, “Setting Global Testing Options.”

Setting Testing Options for a Single Test

Before you record or run a test, you can use the Test Settings dialog box to modify your testing options.

To set testing options for a single test:

1 Choose **Test** > **Settings**.

The Test Settings dialog box opens. It is divided by subject into five tabbed pages.

2 To choose a page, click a tab.

3 Set an option, as described in "Selecting Testing Options for a Single Test," on page 406.

4 To apply your changes and keep the Test Settings dialog box open, click **Apply**.

5 When you are done, click **OK** to save your changes and close the dialog box.

Selecting Testing Options for a Single Test

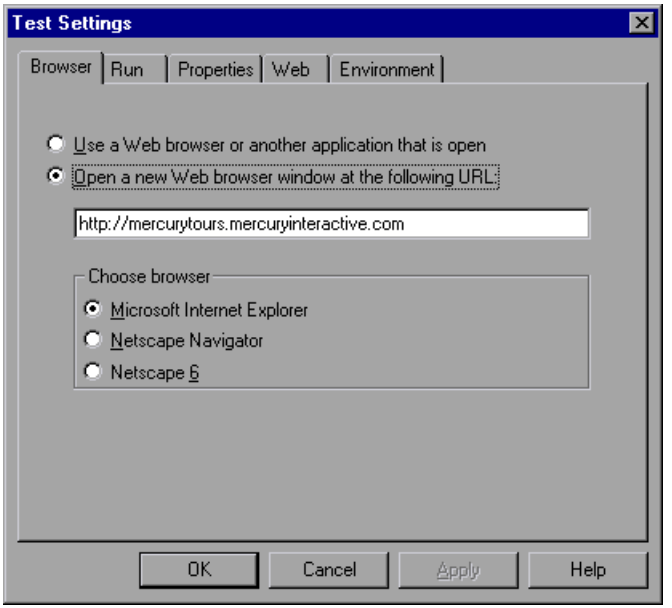
The Test Settings dialog box contains the following tabbed pages:

Tab Heading	Subject
Browser	options for selecting the Web browser and initial URL to use when starting the test
Run	options for setting test iteration preferences
Properties	options for setting the properties of the test
Web	options for setting how the test records and runs on a Web browser
Environment	options for viewing and modifying environment variables, and selecting the active External environment variables file

This section lists the testing options you can set using the Test Settings dialog box.

Browser Test Settings

The Browser tab defines Web browser options. It specifies which browser is to be used during testing. Options include the ability to use a currently running browser or to open a new browser window with a specific location (URL) for testing.



Note: You can also set the browser options in the Browser Settings dialog box, which opens when you start recording.

You can set new browser options for different recording sessions of one test (i.e. when recording a new action). QuickTest retains the new settings for the test the next time you open it. To ensure that your test runs properly, confirm that the browser testing options are set to correctly open the Web page in the first step before you run the test.

For more information about the Browser Settings dialog box, see "Recording a Test," on page 53.

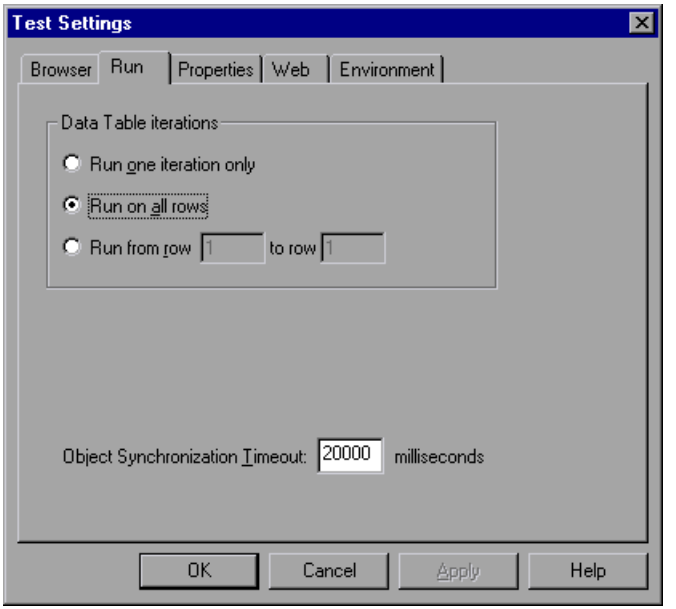
The Browser tab includes the following options:

Option	Description
Use a Web browser or another application that is open	Instructs QuickTest to use your existing browser window or another application that is open to record and run the test. **Tip:** Use this option to modify the starting URL for an existing test. **Note:** QuickTest supports Microsoft Internet Explorer, Netscape Navigator, and AOL. For information on supported browser versions, refer to the *Read Me* file.
Open a new Web browser window at the following URL	Instructs QuickTest to open a new browser session to record and run the test using the specified Web location address. **Note:** QuickTest supports Microsoft Internet Explorer and Netscape Navigator versions with this option. For information on supported browser versions, refer to the *Read Me* file.
Choose browser	Instructs QuickTest to use the specified browser to record and run the test.

Run Test Settings

When you run a test, QuickTest performs the steps you recorded on your Web site. When you run a test with global parameters, by default, QuickTest runs the test for each row in the table in the Data pane, using the parameters you specified. You can choose to run a range of data sets. For more information, see "Setting Data Table Parameters as Global or Action," on page 155.

You can use the Run tab to instruct QuickTest to parameterize a test or an action for only certain lines in the global tab in the Data pane.



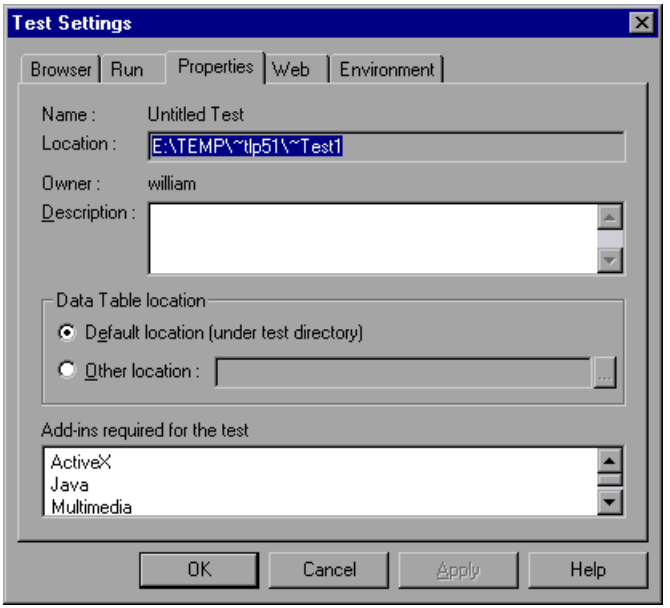
The Run tab includes the following options:

Option	Description
Run one iteration only	Runs the test only once, using only the first row in the global data table.
Run on all rows	Runs the test with iterations using all rows in the global data table
Run from row	Runs the test with iterations using the values in the global data table for the specified row range.
Object Synchronization Timeout	Sets the maximum interval (in milliseconds) that QuickTest waits for a Web object to load in the browser before running a test step.

Note: The Run tab of the Test Settings dialog box applies to the entire test. You can set the run properties for an individual action from the Run tab in the Action Properties dialog box of a selected action. For more information about action run properties, see "Setting the Run Properties for an Action," on page 254.

Properties Test Settings

The Properties tab options define general test information.

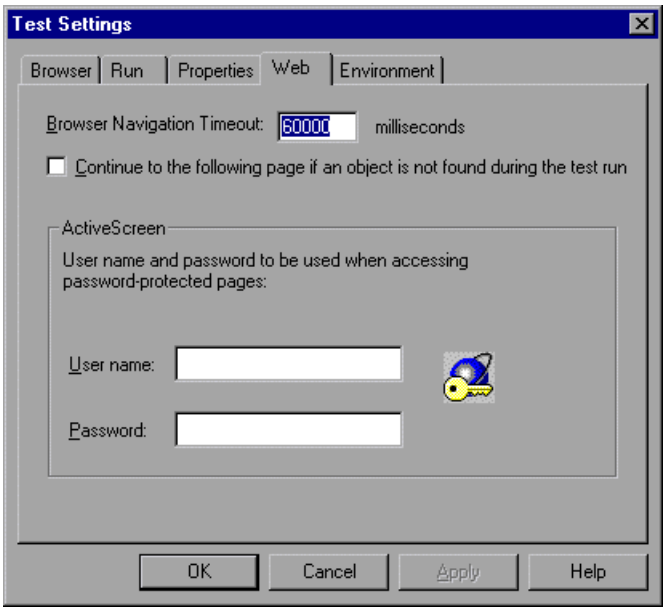


The Properties tab includes the following options:

Option	Description
Name	Indicates the name of the test.
Location	Indicates the path of the test.
Owner	Indicates the user name.
Description	Specifies the test description.
Default location	Instructs QuickTest to use data stored in the default data table location under the test folder.
Other location	Instructs QuickTest to use data stored in the specified data table location. The data table can be any Microsoft Excel (*.xls*) file.
Add-ins required for the test	Displays the add-ins loaded with the Add-in Manager.

Web Test Settings

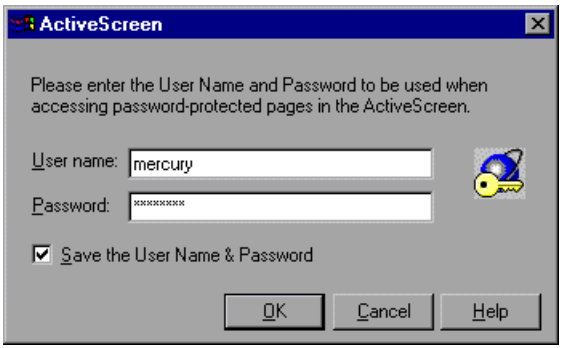
The Web tab options provide additional specifications for recording and running tests on Web pages. For example, if a Web page is password-protected, then so is the corresponding page in the ActiveScreen. You can enter a user name and password for accessing password-protected ActiveScreen pages so that you are not prompted for them each time.



The Web tab includes the following options:

Option	Description
Browser Navigation Timeout	Sets the interval (in milliseconds) QuickTest waits for the Web page to load before running a test step.
Continue to the following page if an object is not found during the test run	Instructs QuickTest not to perform further operations on the current page and to navigate to the next page in the test when an object is not found, which enables the test to continue running. **Note:** When QuickTest implements this behavior after an object is not found, the step is marked with a warning, but the test can still be marked as passed.
User name	The user name for password-protected pages in your test.
Password	The password for password-protected pages in your test.

If you choose not to enter a user name and password in the ActiveScreen box and you attempt to view a page in the ActiveScreen (in your recorded test or in the test results) that requires a password, the ActiveScreen dialog box opens:



You must enter the user name and password in this dialog box in order to view the page. If you want the user name and password to be saved for future use, select the **Save the User Name & Password** check box.

Environment Test Settings

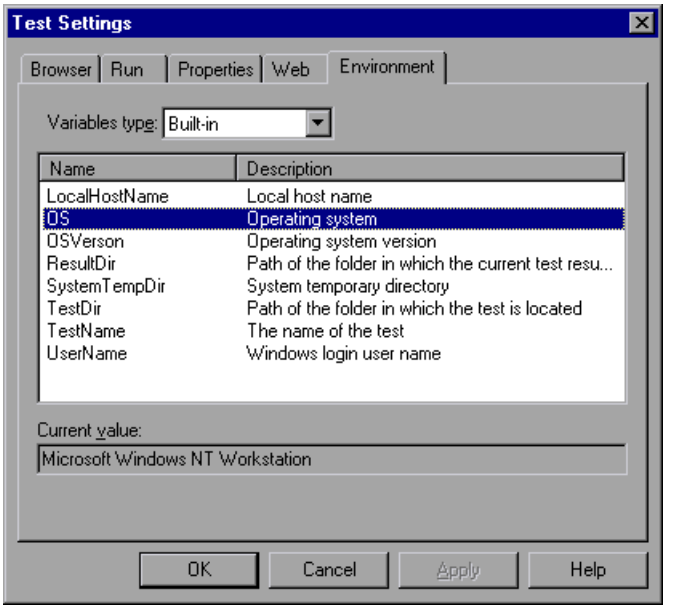
The Environment tab displays existing environment variables and the active external environment variable file. It also enables you to add, modify or delete internal environment variables.

For more information about environment variables and environment parameters, see Chapter 10, "Parameterizing Tests."

The Environment tab includes the following main options:

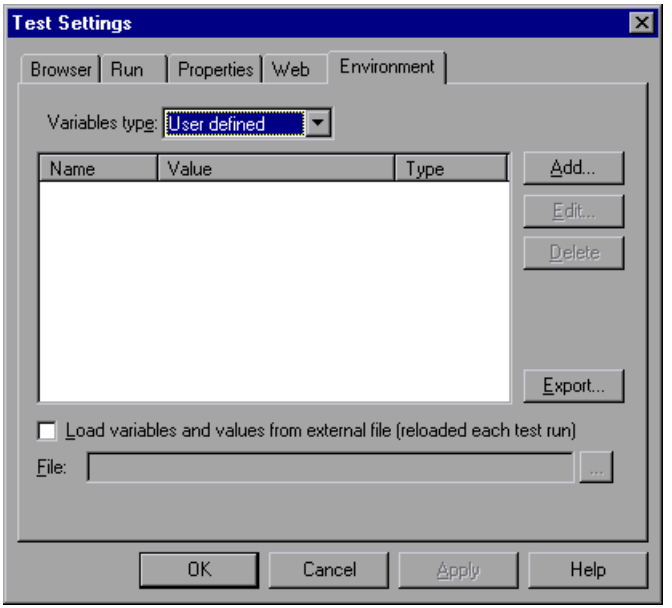
Option	Description
Built-in	Displays built-in environment variables.
User defined	Displays user-defined environment variables.

When “Built-in” is selected, the Environment tab displays the following information:



Option	Description
Name	The name of the built-in variable.
Description	The description of the built-in variable.
Current value	The current value of the selected built-in variable.

When "User defined" is selected, the Environment tab displays the following information:



Option	Description
Name	The user-defined variable name. Variables from the external environment variables file are displayed in blue. Internal environment variables are displayed in black.
Value	The variable value.
Type	The variable type: internal or external.
Add	Opens the Add New Environment Parameter dialog box, enabling you to add a new internal variable.
Edit	Opens the Modify Environment Parameter dialog box, enabling you to edit the selected internal variable.
Delete	Deletes the selected internal variable.

Option	Description
Export	Opens the Save User-Defined Variables dialog box, enabling you to export the current list of User-defined variables and values to a text file. You can then use this file as an external environment variable file.
Load variables and values from external file (reloaded each test run)	Enables the option to load variables and values from the selected external environment variables file.
File	Indicates the selected external environment variables file.

Setting Navigation Fallback Properties

You can use the Navigation Fallback Properties as a backup means of navigating to the next page in the test when both of the following conditions occur:

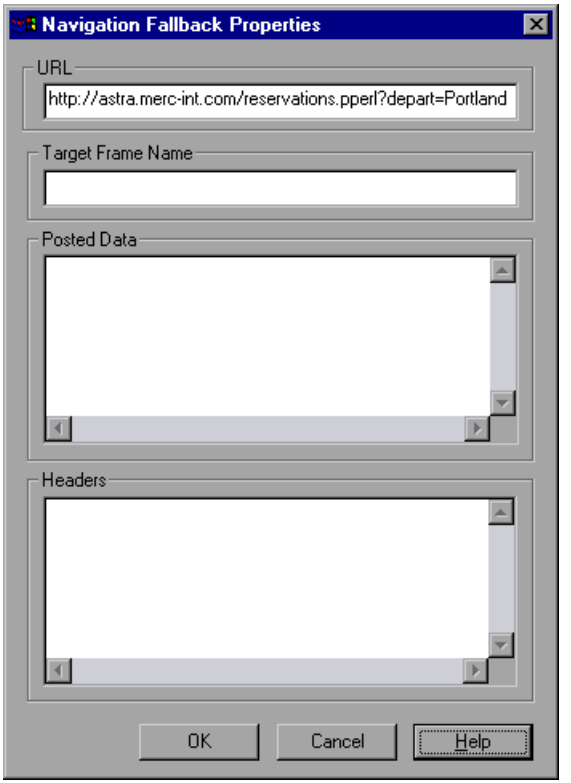
- the navigation step to a page fails during a test run
- the **Continue to the following page if an object is not found during the test run** option is selected in the Web tab of the Test Settings dialog box

You can set the Navigation Fallback Properties for each page in your test.

To set navigation fallback properties:

1 Right-click a page or step icon in the test tree view or in a script line in the Expert view.

2 Select **Navigation Fallback Properties**. The Navigation Fallback Properties dialog box opens.



3 Enter the following page navigation details and click **OK**.

- **URL** - The complete URL of the next page in the test.
- **Target Frame Name** - The name of the frame in which the specified URL should be displayed.
- **Posted Data** - The data to be sent to the server with the HTTP POST transaction. The POST transaction is used to send data gathered by an HTML form to the server. This parameter is ignored if the URL is not an HTTP URL.
- **Headers** - The HTTP header data that is passed to the server upon navigation. For example: Content-Type: application/x-www-form-urlencoded

  This parameter is ignored if the URL is not an HTTP URL.

4 Repeat steps 1 - 3 for each page for which you want to set navigation fallback properties.

26

Customizing the Expert View

You can customize the way your test is displayed when you work in the Expert View. QuickTest includes a powerful and customizable test script editor. You can set the size of margins in the Expert View tab, change the way the elements of a test script are displayed, and create a list of typing errors that will be automatically corrected by QuickTest.

This chapter describes:

- Setting Display Options
- Personalizing Editing Commands

About Customizing Your Test in the Expert View

QuickTest's test script editor lets you set display options, and personalize test script editing commands for working in the Expert View.

Setting Display Options

Display options let you configure the Expert View in QuickTest and customize how your test scripts will be displayed. For example, you can set the size of Expert View tab margins, and activate or deactivate word wrapping.

Display options also let you change the color and appearance of different test script elements. These include comments, strings, QuickTest reserved words, operators and numbers. For each test script element, you can assign colors, text attributes (bold, italic, underline), font, and font size. For example, you could display all strings in the color red.

Finally, there are display options that let you control how the hard copy of your test scripts will be displayed when printed.

Personalizing Test Script Editing Commands

QuickTest includes a list of default keyboard commands that let you move the cursor, delete characters, cut, copy, and paste information to and from the clipboard. You can replace these commands with commands you prefer. For example, you could change the Set Bookmark [#] command from the default CTRL + K + [#] to CTRL + B + [#].

Setting Display Options

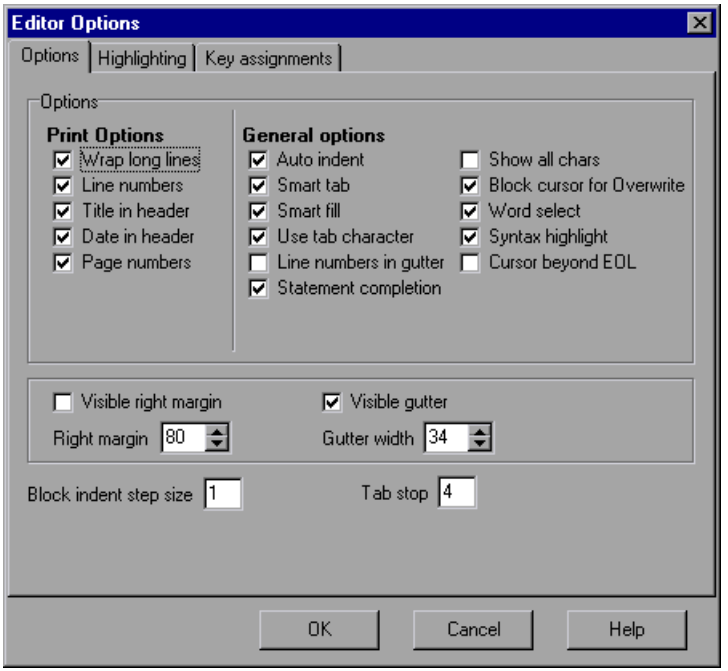
QuickTest's display options let you control how test scripts are displayed in the Expert View tab, how different elements of test scripts are displayed, and how test scripts will be displayed when they are printed.

Customizing Test Scripts

You can customize how QuickTest's test scripts are displayed. For example, you can highlight test script elements and show or hide text symbols.

To customize the appearance of your script:

1 Choose **Tools** > **Editor Options**. The Editor Options dialog box opens.



2 Click the **Options** tab.

3 Under **General options** choose from the following options:

Options	Description
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.
Smart tab	A single press of the tab key will insert the appropriate number of tabs and spaces in order to align the cursor with the text in the line above.

Options	Description
Smart fill	Insert the appropriate number of tabs and spaces in order to apply the Auto indent option. When this option is not selected, only spaces are used to apply the Auto indent. (Both **Auto indent** and **Use tab character** must be selected to apply this option).
Use tab character	Inserts a tab character when the tab key on the keyboard is used. When this option is not enabled, the appropriate number of space characters will be inserted instead.
Line numbers in gutter	Displays a line number next to each line in the script. The line number is displayed in the test script window's gutter.
Statement completion	Opens a list box displaying all available matches to the method prefix whenever the user presses the Ctrl and Space keys simultaneously, the period (.) key, or chooses **Edit > Complete Word**. Select an item from the list to replace the typed string. To close the list box, press the ESC key. Displays a tooltip with the method parameters once the complete method name is displayed in the editor. The method parameters are displayed also whenever the user presses the CTRL, SHIFT, and SPACE keys simultaneously or choose **Edit > Parameter Info**.
Show all chars	Displays all text symbols, such as tabs and paragraph symbols.
Block cursor for Overwrite	Displays a block cursor instead of the standard cursor when you select overwrite mode.
Word select	Selects the nearest word when you double-click in the Expert View tab.
Syntax highlight	Highlights test script elements such as comments, strings, or reserved words.
Cursor beyond EOL	Enables QuickTest to display the cursor after the end of the current line.

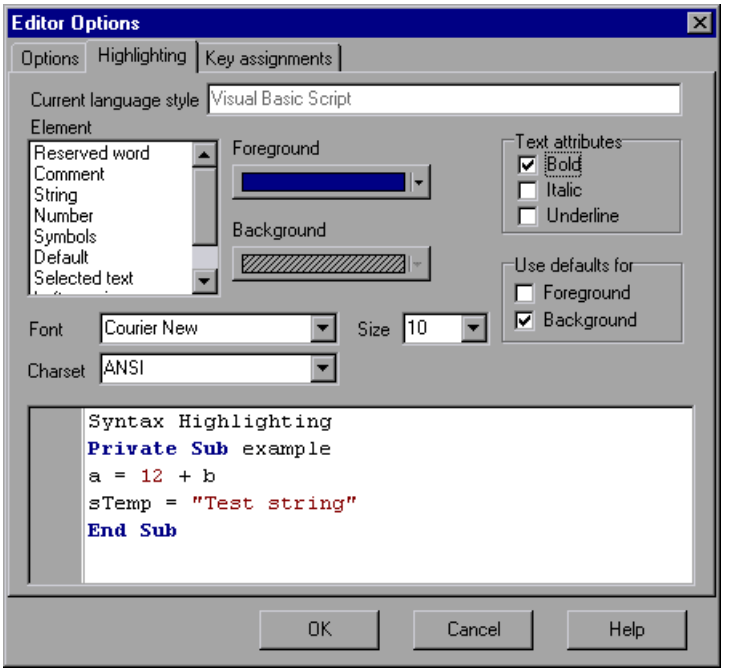
Options	Description
Visible right margin	Displays a line that indicates the Expert View tab's right margin.
Right margin	Sets the position, in characters, of the Expert View tab's right margin (0=left tab edge).
Visible gutter	Displays a blank area (gutter) in the Expert View tab's left margin.
Gutter width	Sets the width, in pixels, of the gutter.
Block indent step size	Sets the number characters that the selected block of VBScript statements will be moved (indented) when the INDENT SELECTED BLOCK softkey is used. For more information on editor softkeys, see "Personalizing Editing Commands," on page 430.
Tab stop	Sets the distance, in characters, between each tab stop.

Highlighting Script Elements

QuickTest test scripts contain many different elements, such as comments, strings, QuickTest reserved words, operators and numbers. Each element of a QuickTest test script is displayed in a different color and style. You can create your own personalized color scheme and style for each script element. For example, all comments in your scripts could be displayed as italicized, blue letters on a yellow background.

To edit script elements:

1 Choose **Tools** > **Editor Options**. The Editor Options dialog box opens to the Options tab. Click the **Highlighting** tab.



2 Select a script element from the **Element** list.

3 Choose from the following options:

Options	Description
Foreground	Sets the color applied to the text of the script element.
Background	Sets the color that is displayed behind the script element.
Text Attributes	Sets the text attributes applied to the script element. You can select bold, italic, or underline or a combination of these attributes.

Options	Description
Use defaults for	Applies the font and colors of the “default” style to the selected style.
Font	Sets the font of the script element.
Size	Set the size, in points, of the script element.
Charset	Sets the character subset of the selected font.

An example of each change you apply will be displayed in the pane at the bottom of the dialog box.

4 Click **OK** to apply the changes.

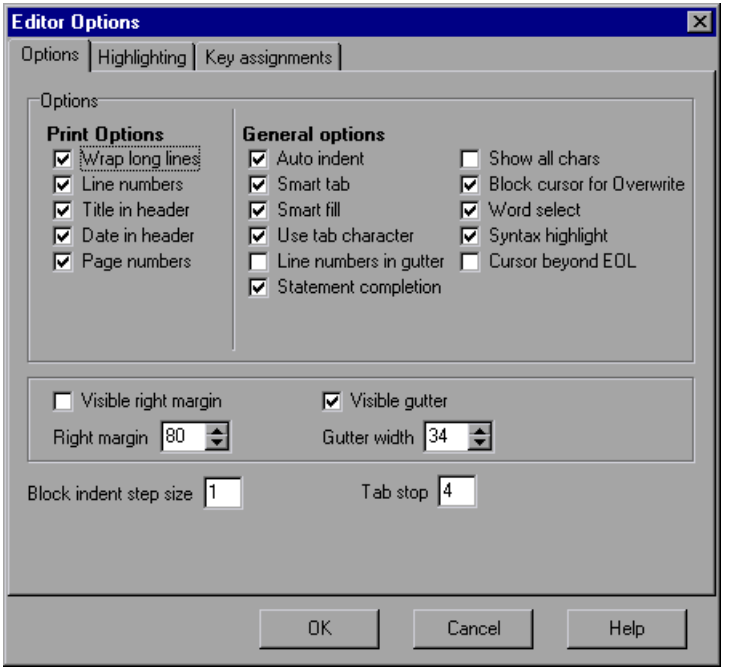
Customizing Print Options

You can set how the hard copy of your script will be displayed when it is sent to the printer. For example, your printed script can include line numbers, the name of the file, and the date it was printed.

To customize your print options:

1 Choose **Tools** > **Editor Options**. The Editor Options dialog box opens.

2 Click the **Options** tab.



3 Choose from the following Print options:

Option	Description
Wrap long lines	Automatically wraps a line of text to the next line if it is wider than the current printer page settings.
Line numbers	Prints a line number next to each line in the script.

Option	Description
Title in header	Inserts the Title into the header of the printed script.
Date in header	Inserts today's date into the header of the printed script.
Page numbers	Numbers each page of the script.

4 Click **OK** to apply the changes.

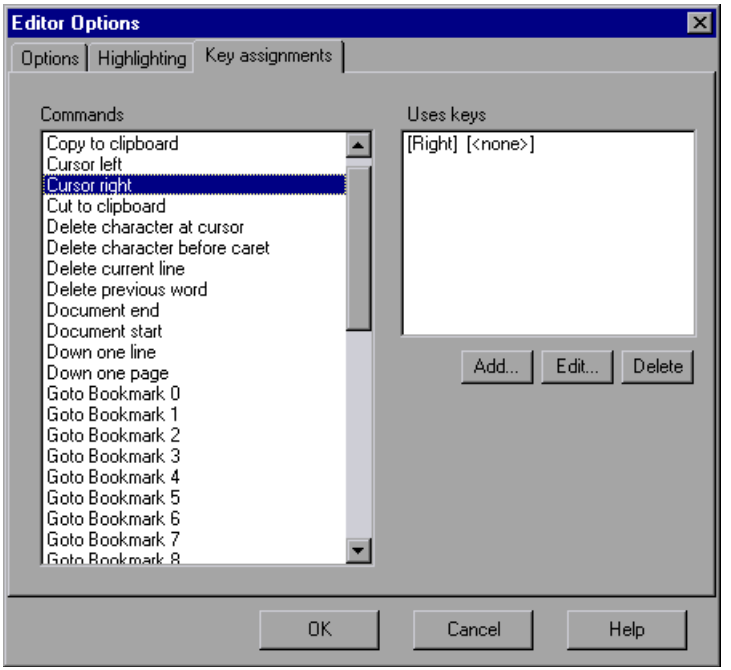
Personalizing Editing Commands

You can personalize the default keyboard commands you use for editing test scripts. QuickTest includes keyboard commands that let you move the cursor, delete characters, cut, copy, and paste information to and from the clipboard. You can replace these commands with your own preferred commands. For example, you could change the Paste command from the default CTRL + V TO CTRL + P.

To personalize editing commands:

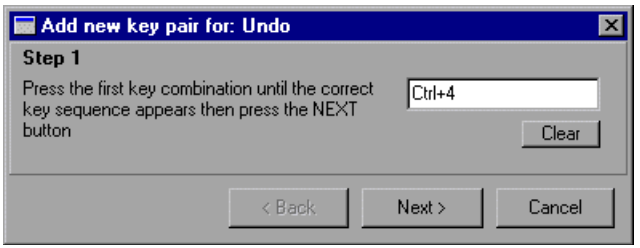
1 Choose **Tools** > **Editor Options**. The Editor Options dialog box opens.

2 Click the **Key Assignments** tab.

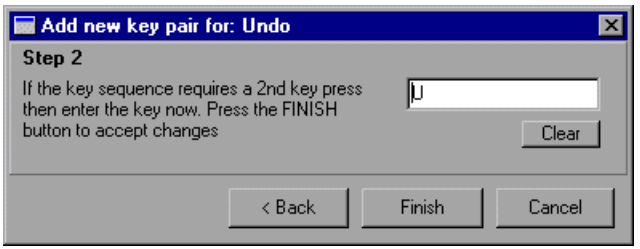


3 Select a command from the **Commands** list.

4 Click **Add** to create an additional key assignment or click **Edit** to modify the existing assignment. The Add/Edit key pair for dialog box opens. Press the keys you want to use. For example, CTRL + 4.



5 Click **Next**. To add an additional key sequence, press the keys you want to use. For example, U.



6 Click **Finish** to add the key sequence(s) to the **Use keys** list.

If you want to delete a key sequence from the list, highlight the keys in the **Uses Key** list and click **Delete**.

7 Click **OK** to apply the changes.

27

Setting Testing Options from a Test Script

You can control how QuickTest records and runs tests by setting and retrieving testing options from within a test script.

This chapter describes:

- Setting Testing Options
- Retrieving Testing Options
- Controlling the Test Run
- Adding and Removing Runtime Settings

About Setting Testing Options from a Test Script

QuickTest testing options affect how you record test scripts and run tests. For example, you can set the maximum time that QuickTest allows for finding an object in a page.

You can set and retrieve the values of testing options from within a test script using the **Setting Object** method in the Expert View. For more information on Programming in the Expert View, see Chapter 22, "Testing in the Expert View."

By retrieving and setting testing options in a test script using the **Setting Object** function, you can control how QuickTest runs a test.

You can also set many testing options using the Options dialog box (global testing options) and the Test Settings dialog box (test-specific settings). For more information on setting global testing options using the Options dialog box, see Chapter 24, "Setting Global Testing Options." For more information on setting options for a single test, see Chapter 25, "Setting Testing Options for a Single Test."

Setting Testing Options

You can use the **Setting Object** function to set the value of a testing option from within the test script. To set the option, use the following syntax:

Setting (*testing_option* **) =** *new_value*

Some options are global and others are per-test settings:

Using the **Setting** object with a global testing option changes a testing option globally, and this change is reflected in the Options dialog box.

For example, if you execute the following statement:

```
Setting("AutomaticLinkRun")=0
```

QuickTest disables automatically created checkpoints in the test. The setting remains in effect until it is changed again, either with another **Setting** statement, or by clearing the **Ignore automatic checkpoints while running tests** check box in the Advanced Web Options dialog box (Choose **Tools** > **Options** > **Web** tab, and click **Advanced**).

Using the **Setting** object to set per-test options is reflected in the Test Settings dialog box. You can also use the Setting object to change a setting for a specific part of a specific test. For more information see "Controlling the Test Run," on page 437.

For example, if you execute the following statement:

```
Setting("WebTimeOut")=50000
```

QuickTest automatically changes the amount of time it waits for a Web page to load before running a test step to 50000 milliseconds. The setting remains in effect until it is changed again, either with another **Setting** statement, or by setting the **Browser navigation timeout** option in the Web tab of the Test Settings dialog box.

The section lists some of the QuickTest testing options that can be used with the Setting object from within a test script. The corresponding dialog box option is listed where applicable.

AutomaticLinkRun

Sets or retrieves the setting for the Ignore automatic checkpoints while running tests option.

AutomaticLinkRun is a global testing option.

Note that you may also set this option using the **Ignore automatic checkpoints while running tests** option in the Advanced Web Options dialog box as described in "Advanced Web Options," on page 399.

DefaultLoadTime

Sets or retrieves the setting for the **Add additional seconds to page load time** option (in seconds).

DefaultLoadTime is a global testing option.

Note that you may also set this option using the **Add additional seconds to page load time** option in the Web tab of the Options dialog box as described in "Web Options," on page 397.

DefaultTimeOut

Sets or retrieves the delay (in milliseconds) for finding objects.

DefaultTimeOut is a test-specific option.

Note that you may also set this option using the **Object synchronization timeout** option in the Run tab of the Test Settings dialog box as described in "Run Test Settings," on page 409.

WebTimeOut

Sets or retrieves the delay (in milliseconds) for navigating to a URL address.

WebTimeOut is a per-test setting.

Note that you may also set this option using the **Browser navigation timeout** option in the Web tab of the Test Settings dialog box as described in "Web Test Settings," on page 412.

Retrieving Testing Options

You can also use the **Setting** object to retrieve the current value of a testing option. To retrieve the value of a testing option, use the following syntax:

Setting (*testing_option* **)**

To store the value in a variable, use the syntax:

new_var = **Setting (** *testing_option* **)**

To display the value in a message box, use the syntax:

MsgBox (Setting (*testing_option* **))**

For example:

```
LinkCheckSet = Setting("AutomaticLinkRun")
```

assigns the current value of the AutomaticLinkRun setting to the user-defined variable LinkCheckSet.

Other examples of testing options that you can use to retrieve a setting are shown in “Setting Testing Options” on page 434.

Controlling the Test Run

You can use the retrieve and set capabilities of the **Setting** object together to control a test run without changing global settings. For example, if you want to change the *DefaultTimeOut* testing option to 5 seconds for objects on one Web page only, insert the following statement after the Web page opens in your test script:

```
‘Keep the original value of the DefaultTimeOut testing option
old_delay = Setting ("DefaultTimeOut")

‘Set temporary value for the DefaultTimeOut testing option
Setting("DefaultTimeOut")= 5000
```

To change back the *DefaultTimeOut* testing option to its original value at the end of the Web page, insert the following statement just before linking to the next page in the script:

```
‘Change the DefaultTimeOut testing option back to its original value.
Setting("DefaultTimeOut")=old_delay
```

Adding and Removing Runtime Settings

In addition to the global and test-specific settings, you can also add, modify, and remove your own runtime settings. These settings are applicable during the test run only.

To add a new runtime setting, use the syntax:

Setting.Add (*testing_option*, *value* **)**

For example, you could create a setting that indicates the name of the current tester and writes the name in the report.

```
Setting.Add (“Tester Name”, “Mark Train”)
Reporter.ReportEvent 1, "Test Run By:", paramcount
```

To modify a runtime setting that has already been initialized, use the same syntax you use for setting any standard setting option:

Setting (*testing_option* **) =** *new_value*

For example:

```
Setting(“Tester Name”)=Alice Wonderlin
```

To remove a runtime setting, use the following syntax:

Setting.Remove (*testing_option* **)**

For example:

```
Setting.Remove ("Tester Name")
```

Part VII

Working with Other Mercury Tools

28

Working with TestDirector

Web site testing typically involves creating and running many tests. TestDirector, Mercury Interactive's test management tool, can help you organize and control the testing process.

This chapter describes:

- Using QuickTest with TestDirector
- Connecting to and Disconnecting from a Project
- Saving Tests to a Project
- Opening Tests in a Project
- Running Tests from TestDirector
- Saving Test Results to a Project
- Running WinRunner Tests from TestDirector
- Calling WinRunner Functions from TestDirector

About Working with TestDirector

TestDirector is a powerful Web-based test management tool that helps you systematically control the testing process. It helps you create a framework and foundation for your testing workflow.

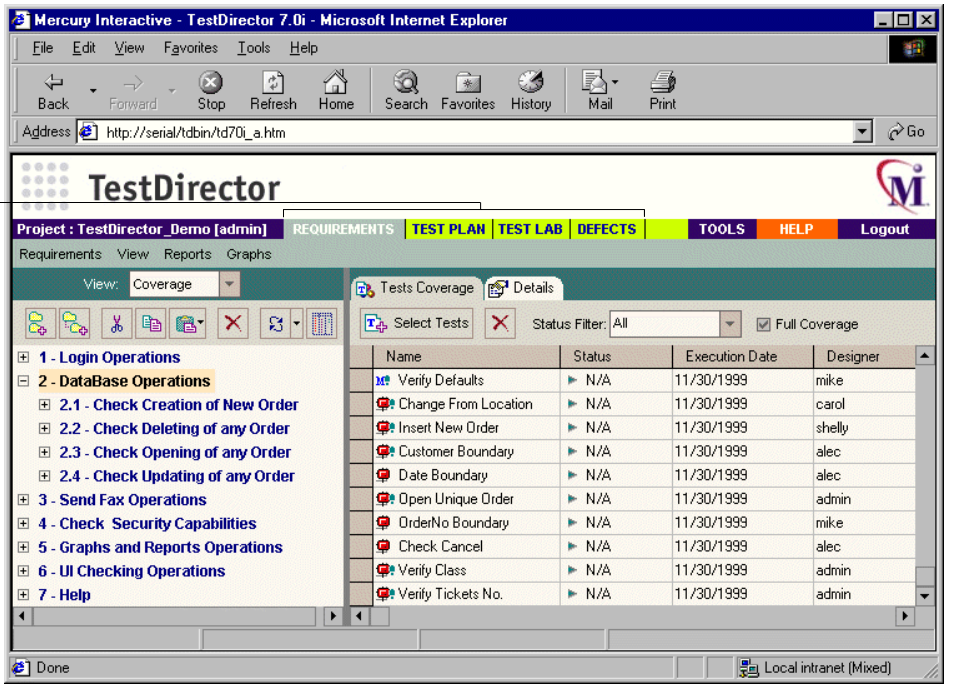
TestDirector helps you maintain a project of tests that covers all aspects of your application's functionality. Every test in your project is designed to fulfill a specified testing requirement of your application. To meet the goals of a project, you organize the tests in your project into unique groups.

TestDirector provides an intuitive and efficient method for scheduling and running tests, collecting test results, and analyzing the results.

It also features a system for tracking defects, enabling you to monitor defects closely from initial detection until resolution.

TestDirector includes four components: Requirements Manager, Test Plan Manager, Test Lab Manager, and Defects Manager.

Modes of operation



The following table describes how you can use each TestDirector Manager:

Manager	Description
Requirements	Specify testing requirements. This includes defining what you are testing, defining requirement topics and items, and analyzing the requirements.
Test Plan	Develop a test plan. This includes defining goals and strategy, dividing your plan into categories, developing tests, and analyzing the plan.
Test Lab	Run tests on your application. This includes defining groups of tests to meet the various testing goals in your project, scheduling test runs, running tests in QuickTest, and analyzing test results.
Defects	Report and track defects. This includes reporting new defects detected in your application, determining repair priorities, repairing open defects, and analyzing the progress of defect repairs.

TestDirector guides you through the requirements specification, test planning, test execution, and defect tracking phases of the testing process. By integrating all the tasks involved in software testing, it helps ensure that your customers receive the highest quality software.

For more information on working with TestDirector, refer to the *TestDirector User's Guide*.

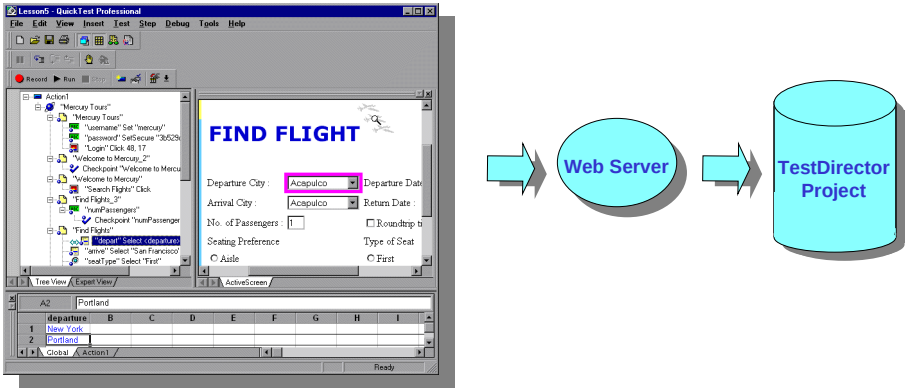
Using QuickTest with TestDirector

Before you start the testing process, you need to create a TestDirector project. A project is a database for collecting and storing data relevant to a testing process. You can create a TestDirector project in Microsoft Access, Oracle, Sybase, or Microsoft SQL.

TestDirector and QuickTest work together to integrate all aspects of the testing process. In QuickTest, you can create tests and save them in your

TestDirector project. After you run your tests, you can view the results in TestDirector.

In order for QuickTest to access the project, you must connect it to the Web server where TestDirector is installed. You can connect to either a local or remote Web server.



QuickTest

When QuickTest is connected to TestDirector, you can:

- save a test by associating it with a subject in the Test Plan manager
- schedule to run a test on local or remote hosts. (Test run results are sent directly to your TestDirector project.)
- report defects to a TestDirector project directly from the Test Results window

For information on reporting defects to a TestDirector project directly from the Test Results window, see Chapter 18, "Analyzing Test Results."

Connecting to and Disconnecting from a Project

If you are working with both QuickTest and TestDirector, QuickTest can communicate with your TestDirector project. You can connect or disconnect QuickTest from a TestDirector project at any time during the testing process. However, do not disconnect QuickTest from TestDirector while a QuickTest test is opened from TestDirector.

The connection process has two stages. First, you connect QuickTest to a local or remote TestDirector Web server. This server handles the connections between QuickTest and the TestDirector project.

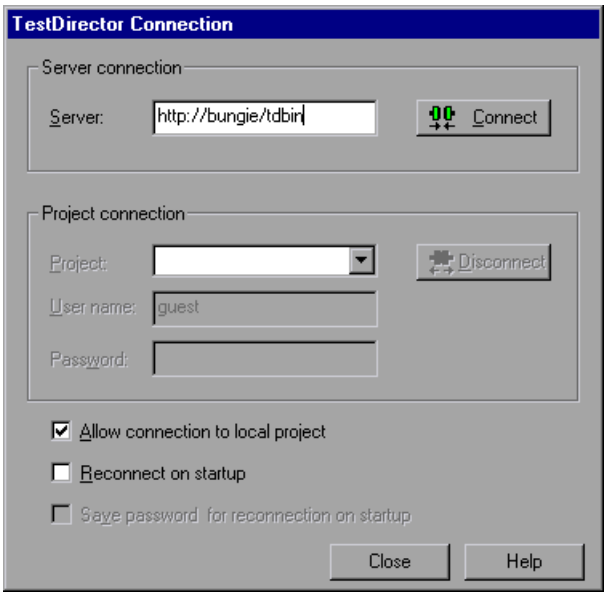
Next, you choose the project you want QuickTest to access. The project stores tests and test run information for the Web site you are testing. Note that TestDirector projects are password protected, so you must provide a user name and a password.

Connecting QuickTest to TestDirector

You must connect QuickTest to the Web server where TestDirector is installed, before you connect QuickTest to a project. For more information, see "Using QuickTest with TestDirector" on page 443.

To connect QuickTest to TestDirector:

1 Choose **Tools** > **TestDirector Connection**. The TestDirector Connection dialog box opens.



2 In the **Server connection** section, in the **Server** box, type the URL address of the Web server where TestDirector is installed.

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

---

3 Click **Connect**.

Once the connection to the server is established, the server's name is displayed in read-only format in the Server box.

4 In the **Project connection** section, select a TestDirector project from the **Project** box.

5 In the **User Name** box, type a user name.

6 In the **Password** box, type a password.

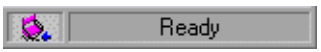
7 Click **Connect** to connect QuickTest to the selected project.

Once the connection to the selected project is established, the project's name is displayed in read-only format in the Project box.

To automatically reconnect to the TestDirector server and the selected project on startup, select the **Reconnect on startup** check box.

If the **Reconnect on startup** check box is selected, then the **Save password for reconnection on startup** check box is enabled. To save your password for reconnection on startup, select the **Save password for reconnection on startup** check box. If you do not save your password, you will be prompted to enter it when QuickTest connects to TestDirector on startup.

8 Click **Close** to close the TestDirector Connection dialog box. The TestDirector icon appears on the status bar to indicate that QuickTest is currently connected to a TestDirector project.



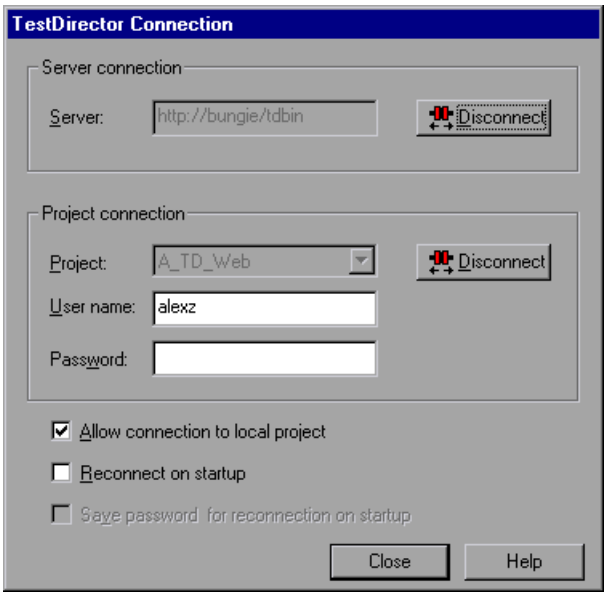
Disconnecting from a TestDirector Project

You can disconnect from a TestDirector project. This enables you to select a different project while using the same server connection.

Note: When disconnecting from TestDirector, if a test is opened from TestDirector, then QuickTest closes it.

To disconnect QuickTest from a TestDirector project:

1 Choose **Tools > TestDirector Connection**. The TestDirector Connection dialog box opens.



2 In the **Project connection** section, click **Disconnect** to disconnect QuickTest from the selected project.

3 Click **Close** to close the TestDirector Connection dialog box.

Disconnecting a TestDirector Server

You can disconnect from a TestDirector server. This enables you to select a different TestDirector server and a different project.

To disconnect QuickTest from a TestDirector server:

1 Choose **Tools** > **TestDirector Connection**.

The TestDirector Connection dialog box opens.

2 In the **Server connection** section, click **Disconnect** to disconnect QuickTest from the TestDirector server.

3 Click **Close** to close the TestDirector Connection dialog box.

Note: If you disconnect QuickTest from a TestDirector server without first disconnecting from a project, QuickTest's connection to that database is automatically disconnected.

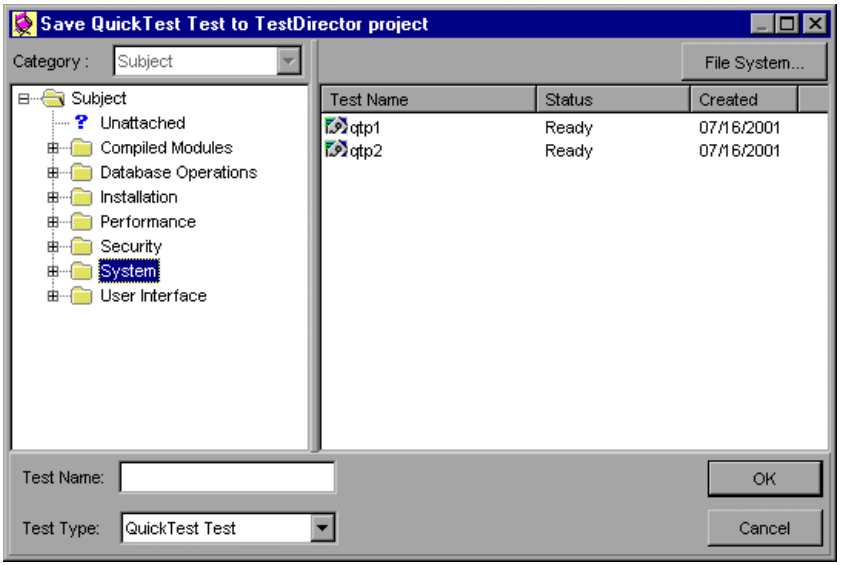
Saving Tests to a Project

When QuickTest is connected to a TestDirector project, you can create new tests in QuickTest and save them directly to your project. To save a test, you give it a descriptive name and associate it with the relevant subject in the test plan tree. This helps you to keep track of the tests created for each subject and to quickly view the progress of test planning and creation.

To save a test to a TestDirector project:

1 In QuickTest, click **Save** or choose **File** > **Save** to save the test.

The Save QuickTest Test to TestDirector Project dialog box opens and displays the test plan tree.



Note that the Save QuickTest Test to TestDirector Project dialog box opens only when QuickTest is connected to a TestDirector project.

To save a test directly in the file system, click the **File System** button to open the Save QuickTest Test dialog box. (From the Save Test dialog box, you may return to the Save Test to TestDirector Project dialog box by clicking the TestDirector button.)

2 Select the relevant subject in the test plan tree. To expand the tree and view a sublevel, double-click a closed folder. To collapse a sublevel, double-click an open folder.

3 In the **Test Name** box, enter a name for the test. Use a descriptive name that will help you easily identify the test.

4 Click **OK** to save the test and close the dialog box. Note that if you are connected to a remote TestDirector project, the word “Uploading” appears in the status bar. This word disappears when QuickTest completes the save test process.

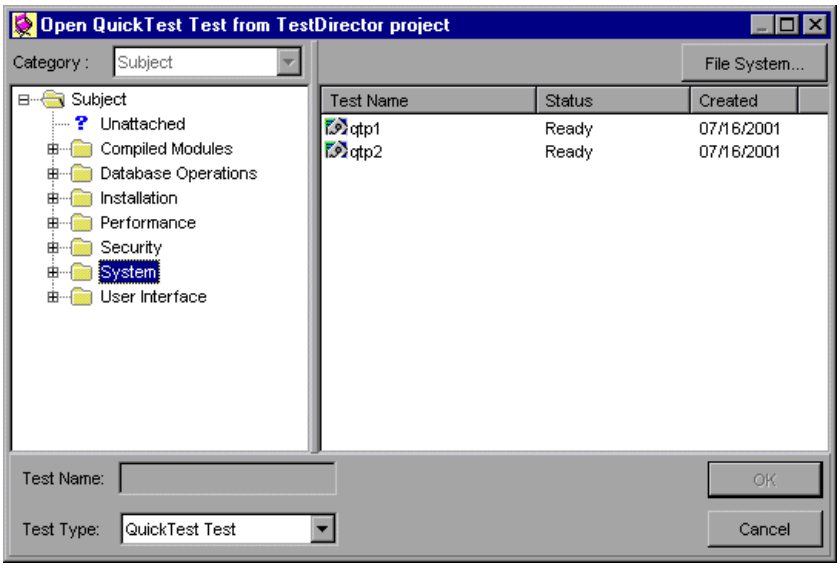
The next time you start TestDirector, the new test will appear in TestDirector's test plan tree. Refer to the *TestDirector User's Guide* for more information.

Opening Tests in a Project

If QuickTest is connected to a TestDirector project, you can open automated tests that are a part of your project. You locate tests according to their position in the test plan tree, rather than by their actual location in the file system.

To open a test saved to a TestDirector project:

1 In QuickTest, click **Open** or choose **File** > **Open** to open the test. The Open QuickTest Test from TestDirector Project dialog box opens and displays the test plan tree.



Note that the Open QuickTest Test from TestDirector Project dialog box opens only when QuickTest is connected to a TestDirector project.

To open a test directly from the file system, click the **File System** button to open the Open QuickTest Test dialog box. (From the Open QuickTest Test

dialog box, you may return to the Open QuickTest Test from TestDirector Project dialog box by clicking the TestDirector button.)

2 Click the relevant subject in the test plan tree. To expand the tree and view sublevels, double-click closed folders. To collapse the tree, double-click open folders.

Note that when you select a subject, the tests that belong to the subject appear in the Test Name list.

3 Select a test in the **Test Name** list. The test appears in the read-only Test Name box.

4 Click **OK** to open the test.

Note that if you are connected to a remote TestDirector project, the word "Downloading" appears in the status bar. This word disappears when the open test process is completed.

When the test opens, the QuickTest title bar displays "[TestDirector]", the full subject path and the test name. For example:

[TestDirector] Subject\System\qa_test1

Note: You can also open tests from the recent tests list in the File menu. If you select a test located in a TestDirector project, but QuickTest is currently not connected to that project, the Connect to TestDirector Project dialog box opens. Enter your user name and password to log in to the project, and click **OK**.

Running Tests from TestDirector

You can run a QuickTest test from a TestDirector project.

Note: If your QuickTest application is installed on Windows 95/98, you must run the Remote Agent on the QuickTest machine before running the test from TestDirector.

To run the Remote Agent, choose **Programs** > **QuickTest** > **Tools** > **Remote Agent** from the **Start** menu. A message box informs you that the Remote Agent is running.

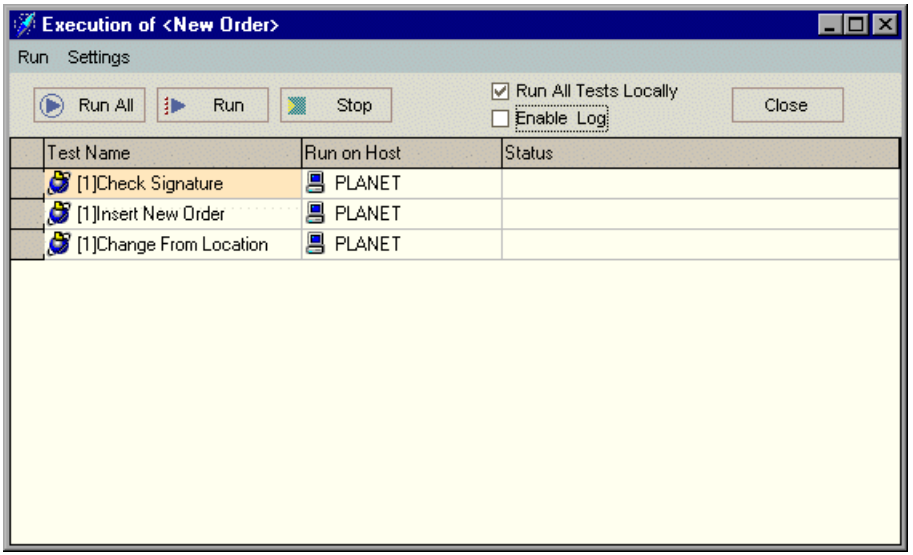
To run a test from a TestDirector project:

1 In TestDirector, click the **Test Lab** tab.

2 You can run all the tests in the test set or select specific tests:

- To run all automated tests in the test set, click the **Run Tests** button or choose **Execution** > **Run All Tests**.
- To run specific automated tests, select the tests and click the **Run Tests** button or choose **Execution** > **Run Selected Tests**.

The Execution dialog box opens and displays the selected tests.



3 You can run your tests locally or remotely:

- Select the **Run All Tests Locally** check box to execute the tests locally.
- Clear the **Run All Tests Locally** check box to execute the tests remotely. To change the designated host for a test, place the mouse pointer in the **Run on Host** grid box, and click the browse button. Select a host from the **Select Host** list, or select a group host from the **Select Host Group** list.

4 Click **Run**. Alternatively, click **Run All** to run all the tests.

Note that test execution will commence only when the selected host becomes available to run tests. TestDirector displays the test execution progress in the Status column. Click **Stop** if you need to terminate test execution before it is complete.

5 After test execution is complete, click **Close** to close the Execution dialog box.

You can view a summary of test results in TestDirector. The updated status for each test run appears in the Execution Grid tab.

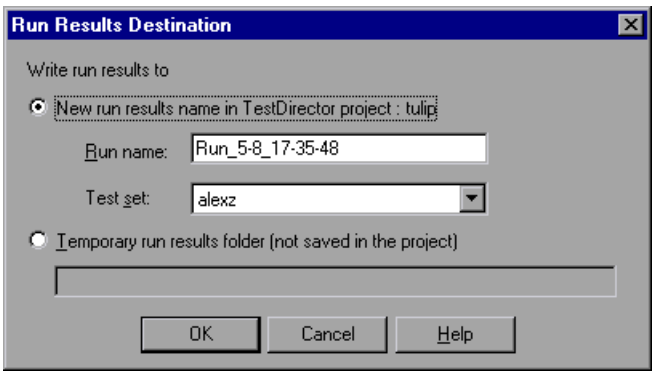
You can see the results of a test run by clicking the Launch Report button in the Test Lab tab in TestDirector.

Saving Test Results to a Project

QuickTest can run a test from a TestDirector project and save the test run results in the project. To save the test run results, you specify a name for the test run and a test set in which to store the results.

To save test run results to a TestDirector project:

1 In QuickTest, click the **Run** button or choose **Test** > **Run**. The Run Results Destination dialog box opens.



2 To save the test run results in the project, accept the default run name, or type a different one in the box. Accept the default test set, or browse to select another one.

Note: A test set is a group of tests selected to achieve specific testing goals. For example, you can create a test set that tests the user interface of the application or the application's performance under stress. You define test sets when working in TestDirector's test run mode.

To run the test and overwrite the previous test run results, Click **Temporary run results folder**.

Note: QuickTest stores temporary test run results for all tests in <*System Drive:**Temp**TempResults*>. The path in the text box of the **Temporary run results folder** option is read-only and cannot be changed.

---

3 Click **OK**. The Run Results Destination dialog box closes and QuickTest begins running the test. As QuickTest runs the test, it highlights each step in the test tree.

When the test stops running, the Test Results window opens unless you have cleared the **View results when test run ends** check box in the Run tab of the Options dialog box. For more information about the Options dialog box, see Chapter 24, "Setting Global Testing Options."

If you run a test located in a remote TestDirector project, when the test stops running, "Uploading results" appears in the status bar. The Test Results window opens when the uploading process is completed.

Note: If you want to interrupt a test that is running, you can:

Click the **Pause** button or choose **Debug** > **Pause**. The test run pauses. To resume running a paused test run, click the **Run** button or choose **Test** > **Run**.

Click the **Stop** button or choose **Test** > **Stop**. The test run stops running and the **Test Results** window opens.

Running WinRunner Tests from TestDirector

When QuickTest is connected to a TestDirector project that contains WinRunner tests, you can run a WinRunner test in this project from TestDirector. You can also open the WinRunner Test Results window for additional details. For additional information on working with WinRunner, see Chapter 29, "Working with WinRunner."

Note: You can also call a WinRunner function that is saved in a compiled module in a TestDirector project. For information on calling a WinRunner function, see "Calling WinRunner Functions from TestDirector" on page 459.

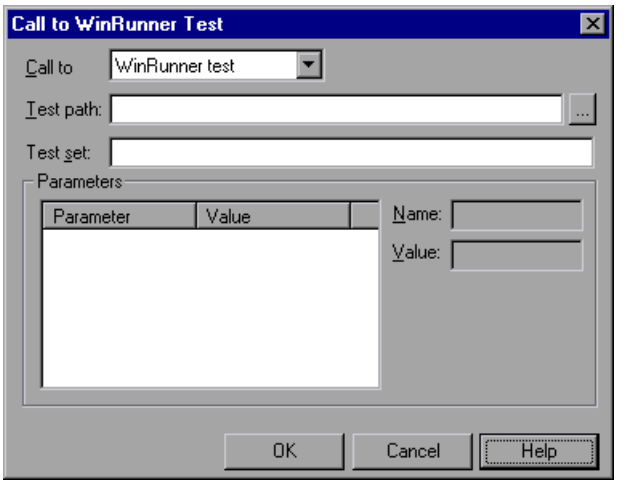
Running a WinRunner Test from a TestDirector Project

To run a WinRunner test from a TestDirector project in QuickTest, you must call it from a QuickTest test.

To run a WinRunner test:

1 Choose **Insert** > **Call to WinRunner Test**.

The Call to WinRunner Test dialog box opens.



2 In the **Test path** box, enter the path of the WinRunner test or browse to it.

3 In the **Test set** box, enter the name of the test set in the TestDirector project. Note that if the test set is not specified, it is set to "default," which exists for each test.

4 The Parameters box lists any test parameters included in the WinRunner test. To update values for the parameters:

- Highlight the parameter in the **Parameters** pane.
- Enter the new value in the **Value** box.

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

For information on WinRunner test parameters, refer to the *WinRunner User's Guide*.

In QuickTest, the link to the WinRunner test is displayed as:

- a WinRunner **RunTest** item in the Tree View. For example:

- a **TSLTest.RunTest** statement in VBScript in the Expert View. This has the following syntax:

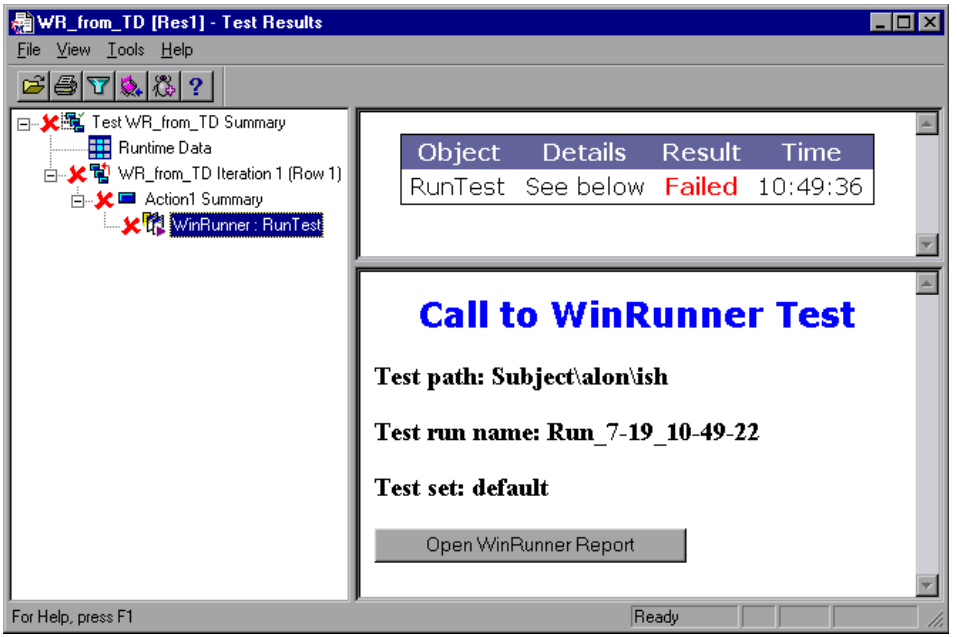
  TSLTest.RunTest (*TestPath* **,** *TestSet* **[,** *Parameters* **])**

Note: If you do not specify a TestSet, the WinRunner test run may fail.

For additional information and an example of usage, refer to the *QuickTest Object Model Reference*.

Viewing the Results

After you run your test in QuickTest, highlight the **WinRunner.RunTest** item in the test results tree to display additional information about the test run.



To view the test results in the WinRunner Test Results window, click **Open WinRunner Report**. For additional information about viewing WinRunner test results, refer to the *WinRunner User's Guide*.

Calling WinRunner Functions from TestDirector

When QuickTest is connected to a TestDirector project that contains WinRunner compiled modules, you can call a TSL function in that compiled module from within a QuickTest test. This is useful when you want to use a user-defined function in WinRunner in aQuickTest test.

Calling a WinRunner Function

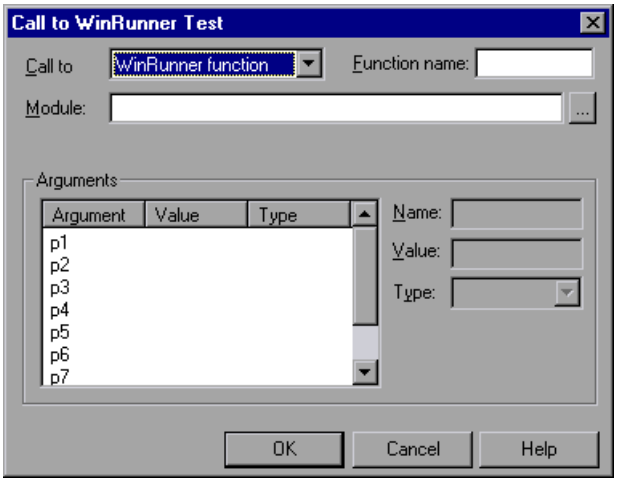
You can call a WinRunner (TSL) function from a QuickTest test by specifying the function and the compiled module containing the function.

To call a TSL function from a WinRunner compiled module:

1 Choose **Insert** > **Call to WinRunner Test**.

The Call to WinRunner Test dialog box opens.

2 In the **Call to** box, choose "WinRunner function".



3 In the **Function name** box, enter the name of the function to call.

4 In the **Module** box, enter the path of the compiled module containing the function or browse to it.

5 The Arguments box lists eight possible arguments for the TSL function. Note that this is the maximum number of arguments that can be sent to a TSL function. To update values for the arguments:

- Highlight the argument in the **Arguments** pane. Note that the Argument column lists p1-p8, which represent arguments in the TSL function in sequential order.
- In the **Type** box, choose the argument type (in/out/inout).
- If the argument type is "in" or "inout," enter the new value in the **Value** box.

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

For information on TSL functions, WinRunner compiled modules and test parameters, refer to the *WinRunner User's Guide*.

In QuickTest, the call to the TSL function is displayed as:

- a WinRunner **CallFunc** item in the Tree View. For example:

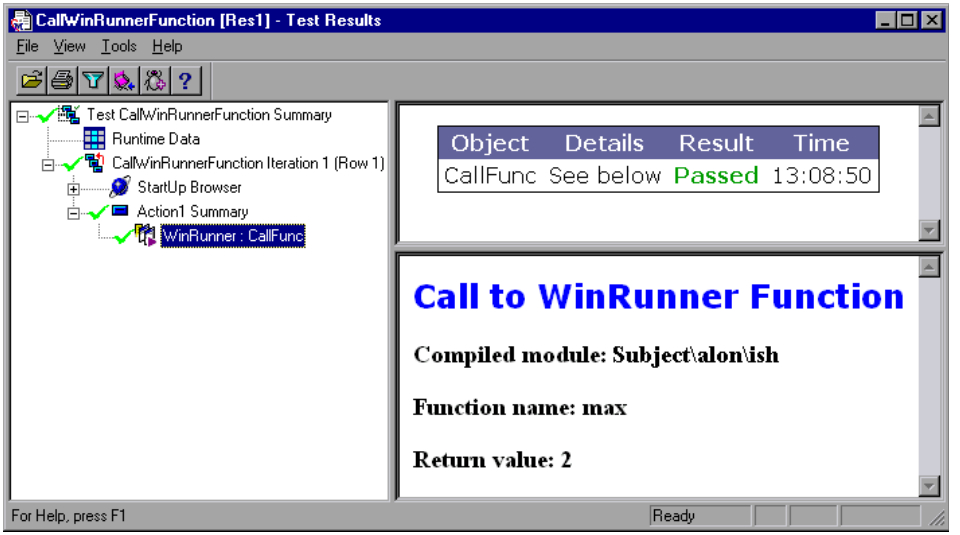
- a **TSLTest.CallFunc** statement in VBScript in the Expert View. This has the following syntax:

TSLTest.CallFunc (*ModulePath*, *TSLFunction* **[,** *Parameters* **])**

For additional information and an example of usage, refer to the *QuickTest Object Model Reference*.

Viewing the Results

After you run your test in QuickTest, highlight the **WinRunner.RunFunc** item in the test results tree to display additional information about the call to the function.



29

Working with WinRunner

When you work with QuickTest, you can also run WinRunner tests and call TSL functions in compiled modules.

This chapter describes:

- Running WinRunner Tests
- Calling WinRunner Functions

About Working with WinRunner

When you run QuickTest tests, you can also run WinRunner tests and call TSL functions in compiled modules with parameters by creating calls to WinRunner tests and functions.

Note: This feature is supported for WinRunner versions 7.01 and higher.

Once you create a call to a WinRunner test or function, you can modify parameter values in existing call statements by editing them in the Expert View. For information on working in the Expert View, see Chapter 22, "Testing in the Expert View."

Note: You cannot run WinRunner tests on Web pages (using WinRunner's WebTest Add-in) from QuickTest.

Running WinRunner Tests

When QuickTest links to WinRunner to run a test, it starts WinRunner, opens the test, and runs it. Information about the test results is displayed in the Test Results window. You can also open the WinRunner Test Results window for additional details.

Note for TestDirector users: For information about running a WinRunner test stored in a TestDirector project, see Chapter 28, "Working with TestDirector."

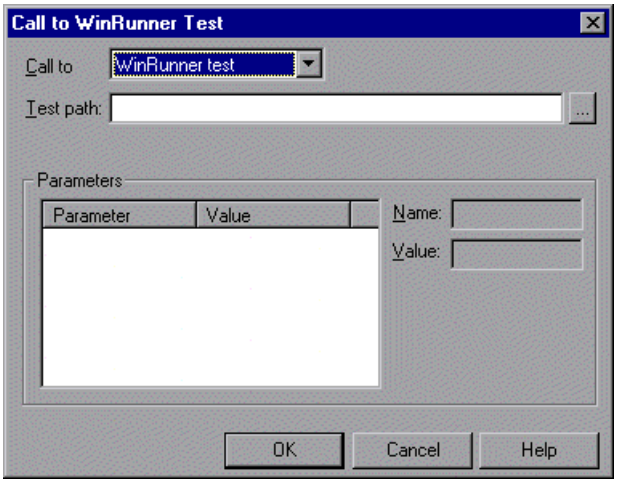
Running a WinRunner Test

To run a WinRunner test from QuickTest, you must call it from a QuickTest test.

To run a WinRunner test:

1 Choose **Insert** > **Call to WinRunner Test**.

The Call to WinRunner Test dialog box opens.



2 In the **Test path** box, enter the path of the WinRunner test or browse to it.

3 The Parameters box lists any test parameters included in the WinRunner test. To update values for the parameters:

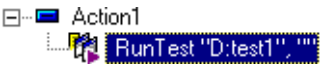
- Highlight the parameter in the **Parameters** pane.
- Enter the new value in the **Value** box.

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

For information on WinRunner test parameters, refer to the *WinRunner User's Guide*.

In QuickTest, the link to the WinRunner test is displayed as:

- a WinRunner **RunTest** item in the Tree View. For example:



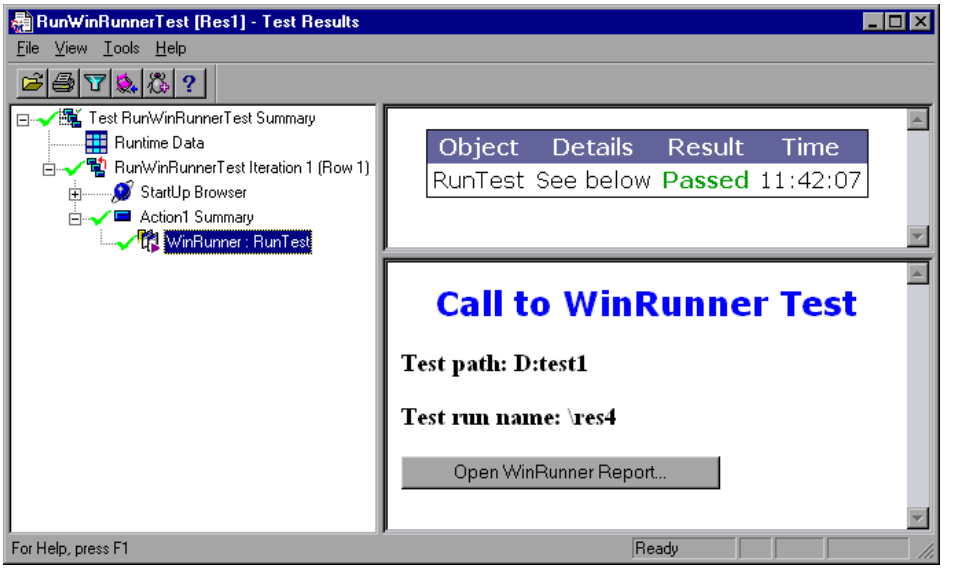
- a **TSLTest.RunTest** statement in VBScript in the Expert View. This has the following syntax:

TSLTest.RunTest (*TestPath* **,** *TestSet* **[,** *Parameters* **])**

For additional information and an example of usage, refer to the *QuickTest Object Model Reference*.

Viewing the Results

After you run your test in QuickTest, highlight the **WinRunner.RunTest** item in the test results tree to display additional information about the test run.



To view the test results in the WinRunner Test Results window, click **Open WinRunner Report**. For additional information about viewing WinRunner test results, refer to the *WinRunner User's Guide*.

Calling WinRunner Functions

When QuickTest links to WinRunner to call a TSL function, it starts WinRunner, loads the compiled module, and calls the TSL function. This is useful when you want to use user-defined function in WinRunner in QuickTest.

Note for TestDirector users: You can call a WinRunner function that is saved in a compiled module in a TestDirector project. For additional information, see Chapter 28, "Working with TestDirector."

Calling a WinRunner Function

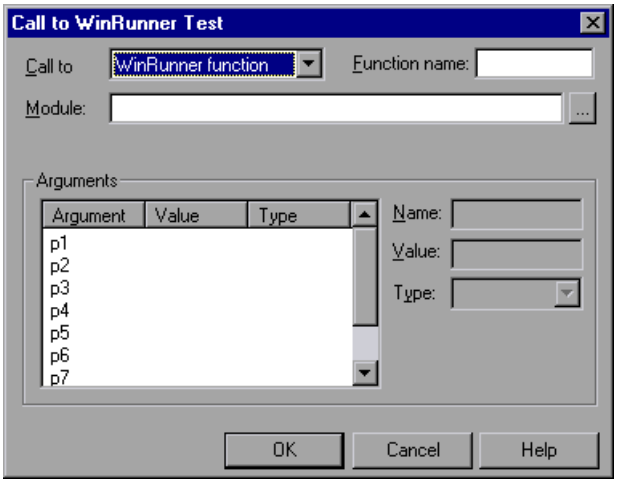
You can call a WinRunner (TSL) function from QuickTest by specifying the function and the compiled module containing the function.

To call a TSL function from a WinRunner compiled module:

1 Choose **Insert** > **Call to WinRunner Test**.

The Call to WinRunner Test dialog box opens.

2 In the **Call to** box, choose "WinRunner function".



3 In the **Function name** box, enter the name of the function to call.

4 In the **Module** box, enter the path of the compiled module containing the function or browse to it.

5 The Arguments box lists eight possible arguments for the TSL function. Note that this is the maximum number of arguments that can be sent to a TSL function. To update values for the arguments:

- Highlight the argument in the **Arguments** pane. Note that the Argument column lists p1-p8, which represent arguments in the TSL function in sequential order.
- In the **Type** box, choose the argument type (in/out/inout).
- If the argument type is "in" or "inout," enter the new value in the **Value** box.

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

For information on TSL functions, WinRunner compiled modules and test parameters, refer to the *WinRunner User's Guide*.

In QuickTest, the call to the TSL function is displayed as:

- a WinRunner **CallFunc** item in the Tree View. For example:

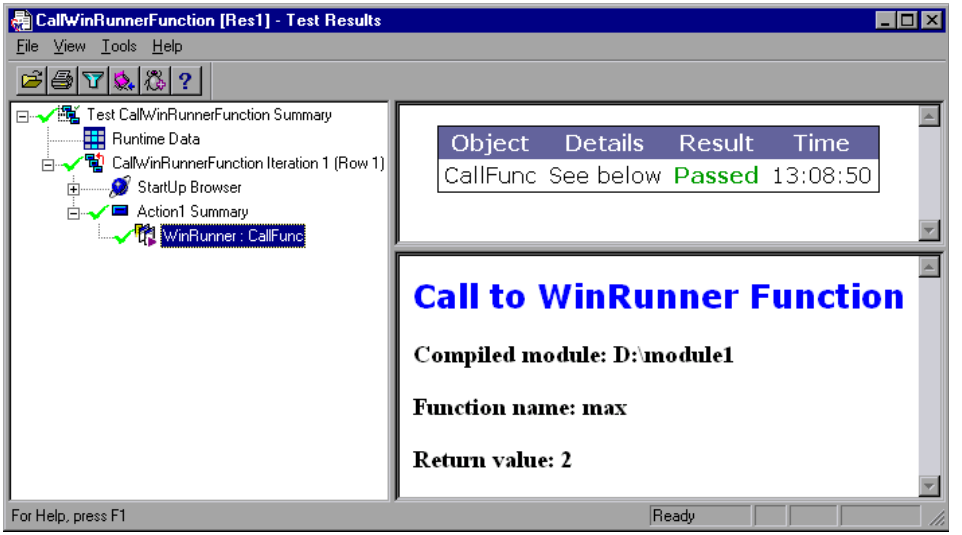
- a **TSLTest.CallFunc** statement in VBScript in the Expert View. This has the following syntax:

TSLTest.CallFunc (*ModulePath*, *TSLFunction* **[,** *Parameters* **])**

For additional information and an example of usage, refer to the *QuickTest Object Model Reference*.

Viewing the Results

After you run your test in QuickTest, highlight the **WinRunner.RunFunc** item in the test results tree to display additional information about the call to the function.



Part VIII

Supported Environments

30

Testing ActiveX Controls

QuickTest supports testing on ActiveX controls in Microsoft Internet Explorer. For information on supported browser versions, refer to the *Read Me* file.

This chapter describes:

- Recording and Running Tests on ActiveX Controls
- Checking ActiveX Controls
- Activating an ActiveX Control Method
- Retrieving and Setting the Values of Properties of ActiveX Controls
- Using Scripting Methods with ActiveX Controls

About Testing ActiveX Controls

Many Web sites include ActiveX controls developed by third-party organizations. QuickTest can run and record tests on these controls as well as check their properties.

QuickTest recognizes ActiveX controls and treats them as it treats standard GUI objects. You can check the properties of an ActiveX control as you check the properties of any other object. For information on creating checkpoints, see Chapter 6, "Creating Checkpoints."

In the Expert View, or using the **Insert** > **Step** option, you can activate ActiveX control methods, retrieve and set the values of properties (including runtime properties), and check that objects exist.

Note that this chapter provides examples using both the Tree View and the Expert View. Some of the information provided and procedures described require working in the Expert View. For information on working in the Expert View, see Chapter 22, "Testing in the Expert View."

Note that you can check ActiveX controls that are tables in the same way that you can check tables in Web pages. For additional information, see "Checking Tables" on page 113.

Recording and Running Tests on ActiveX Controls

QuickTest records and runs steps on ActiveX controls as it does on any other objects. After running your test, QuickTest displays the details of the steps in the Test Results window. For more information on running tests, see Chapter 17, "Running Tests." For more information on viewing test results, see Chapter 18, "Analyzing Test Results."

QuickTest records clicks inside the ActiveX control. When you record on an ActiveX control, QuickTest records an ActiveX icon next to the step in the Tree View. For example, if you record clicking on a calendar that is an ActiveX control, the Tree View may be displayed as follows:

QuickTest records this step in the Expert View as:

```
Browser("Untitled").Page("Untitled").ActiveX("Calendar")
    .Click 65,107
```

Recording on an ActiveX control has the following syntax in the Expert View:

For windowed controls -

...ActiveX (*ActiveX_control* **).Function (** *function_parameters* **)**

For windowless controls -

...wesActiveX (*ActiveX_control* **).Function (** *function_parameters* **)**

QuickTest can record on standard controls within an ActiveX control. For example, suppose your ActiveX control is a calendar that contains a drop-down list from which you can choose the month. If you click in the list to select the month May, QuickTest records this in the Tree View as follows:

QuickTest records this step in the Expert View as:

```
Browser("Untitled").Page("Untitled").ActiveX("Calendar").WinComboBox
    ("ComboBox").Select "May"
```

Note: If an ActiveX control contains another ActiveX control, then QuickTest can run and record tests on this internal control as well.

You run tests on ActiveX controls just as you would with any other QuickTest test. The test results tree displays the same ActiveX control icon as that used in the test tree. The bottom right pane of the Test Results window displays the ActiveX control that was captured during the test run, and highlights the ActiveX control for each step in the test results tree.

For more information about running tests, see Chapter 17, “Running Tests.”

For more information about test results, see Chapter 18, “Analyzing Test Results.”

Checking ActiveX Controls

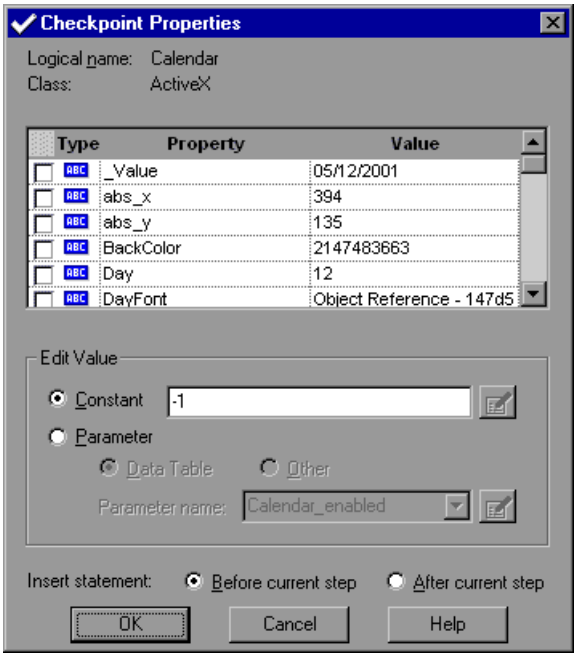
You create a checkpoint on an ActiveX control as you create a checkpoint on any standard Web object. When you create a checkpoint on an ActiveX control, QuickTest captures the ActiveX control properties and their values as it does for any standard Web object. The properties you can check for an ActiveX control depend on the properties of the ActiveX control. By default, when you create a checkpoint on an ActiveX control, QuickTest captures all the properties for an ActiveX control, but it does not select any properties to check.

For example, you can create a checkpoint on an ActiveX control that is a calendar to check the current date in the calendar.

To create a checkpoint on an ActiveX control:

1 After recording your test, right-click the ActiveX control you want to check in the ActiveScreen and select **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.

2 Select the ActiveX control for which you want to create a checkpoint and click **OK**. The Checkpoint Properties dialog box opens and displays all the properties for the ActiveX control.



3 Select the properties you want to check in the check box column.

4 For each property, select the checkpoint options you want to apply.

Click **OK**. A tree item with an ActiveX checkpoint icon is added to your test tree.

For more information on creating checkpoints, see Chapter 6, “Creating Checkpoints.”

Activating an ActiveX Control Method

In the Expert View, you can use the "Object" property to activate the method for an ActiveX control. Activating the method for an ActiveX control has the following syntax:

...ActiveX (*ActiveX_control* **).Object.***Method_to_activate***( )**

For example, suppose the **MakeObjVisible** method is supported for your ActiveX control. To activate the **MakeObjVisible** method, you insert the following statement into your test script:

```
Browser("Home").Page("HomePage").ActiveX("Calendar")
    .Object.MakeObjVisible()
```

For additional information about the "Object" property (.Object), see "Accessing Runtime Object Properties and Methods" on page 374.

Retrieving and Setting the Values of Properties of ActiveX Controls

You can retrieve and set the values of properties of an ActiveX control using the "Object" property.

For example, suppose your ActiveX control is a calendar object. When you record the action of choosing a date in the calendar, your test tree might look like the following:

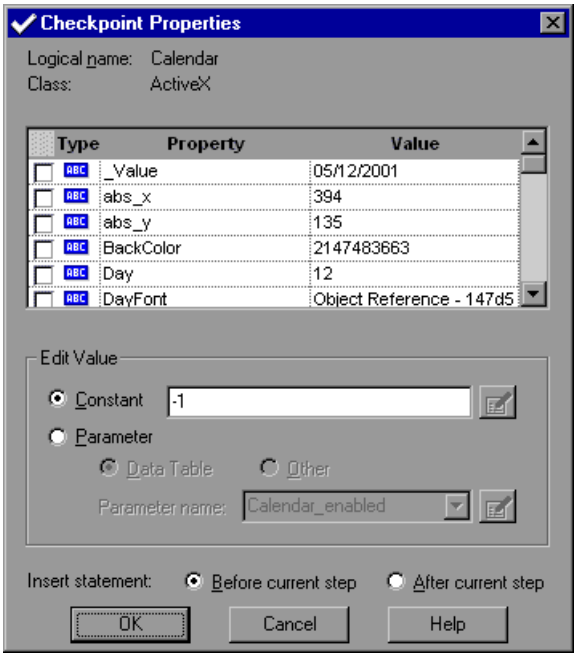
In the same example, the following test script is recorded in VBScript in the Expert View:

```
Browser("Untitled").Page("Untitled").ActiveX("Calendar")
    .WinComboBox("ComboBox").Select "May"
```

```
Browser("Untitled").Page("Untitled").ActiveX("Calendar")
    .Click 152,109
```

Suppose you want to retrieve the date in the calendar, and then you want to change the date. Before you can retrieve and set the date, however, you need to know the name of the corresponding property in your calendar.

You can view the list of properties for the ActiveX control by right-clicking the object in the ActiveScreen and choosing Insert Checkpoint or Insert Output Value. The Checkpoint Properties dialog box opens for the selected object and displays a list of properties that can be retrieved or set.



Now you can use the “Object” property to retrieve and set the day of the month in the calendar, using the Day property.

For additional information about the “Object” property (.Object), see “Accessing Runtime Object Properties and Methods” on page 374.

Retrieving the Value of a Property of an ActiveX Control

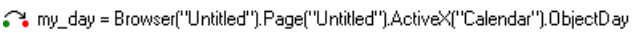
You create a statement in the Expert View using the Object object to retrieve the value of a property of an ActiveX control. The statement has the following syntax:

value **= Browser(***BrowserName***).Page(***PageName***).ActiveX(***ActiveX_control***).**
Object.*ActiveX_control_property*

In the example above, you insert the following statement into your test in the Expert View:

```
my_day = Browser("Untitled").Page("Untitled").
    ActiveX("Calendar").Object.Day
```

This is displayed in the Tree View as:



In the Expert View, you can also add a comment to your script and use the **MsgBox** VBScript method to send a message about the retrieved value. Alternatively, you could write this information to the data table or the test report.

Setting the Value of a Property of an ActiveX Control

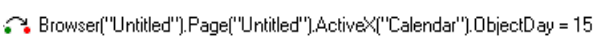
You create a statement in the Expert View using the Object object to set the value of a property of an ActiveX control. The statement has the following syntax:

Browser(*BrowserName***).Page(***PageName***).ActiveX(***ActiveX_control***).**
Object.*ActiveX_control_property* **=** *value*

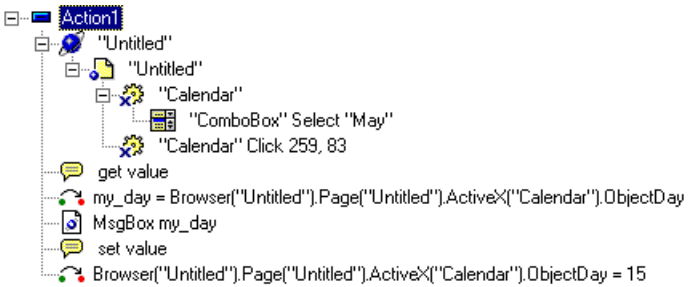
In the example above, you insert the following statement into your test in the Expert View:

```
Browser("Untitled").Page("Untitled").ActiveX("Calendar")
    .Object.Day = 15
```

This is displayed in the Tree View as:



When you are done, your test is displayed as follows in the Tree View:



It is presented as follows in the Expert View:

```
Browser("Untitled").Page("Untitled").ActiveX("Calendar")
    .WinComboBox("ComboBox").Select "May"
Browser("Untitled").Page("Untitled").ActiveX("Calendar").Click 259,83
'get value
my_day=Browser("Untitled").Page("Untitled").ActiveX("Calendar")
    .Object.Day
MsgBox my_day
'set value
Browser("Untitled").Page("Untitled").ActiveX("Calendar").Object.Day=15
```

Using Scripting Methods with ActiveX Controls

QuickTest provides several scripting methods that you can use with ActiveX controls. You can record some of these methods while recording on ActiveX controls. You can enter statements manually with the other methods in the Expert View. For more information about programming in the Expert View, see Chapter 22, "Testing in the Expert View."

For additional information on these methods, refer to the *QuickTest Object Model Reference*.

31

Testing Java Applets and Applications

You can record and run tests on standalone Java applets and applications. For information on supported Java toolkits and versions, refer to the *Read Me* file.

Note: Testing on Java objects is supported only in QuickTest Professional.

This chapter describes:

- Recording and Running Tests on Java Applets and Applications
- Checking Java Applets, Applications, and Objects
- Retrieving and Setting Java Test Settings
- Working with Objects, Methods, and Events in Your Java Applet or Application
- Adding Java Statements in the Expert View

About Testing Java Applets and Applications

In QuickTest, you can record and run steps on Java objects in Internet Explorer or Netscape, in Sun's AppletViewer, and in standalone Java applets or applications. QuickTest records user actions on applets and applications and on the standard Java objects within them.

You can create checkpoints on Java applets, applications, or objects, just as you would for any Web object.

You can also create or enhance tests by entering scripting statements in the Expert View. Java applets, applications and objects have Java-specific object names and methods. For information on working in the Expert View, see Chapter 22, "Testing in the Expert View."

Note that you can check Java table objects in the same way that you check tables in Web pages. For additional information, see "Checking Tables" on page 113.

Recording and Running Tests on Java Applets and Applications

Once you have installed QuickTest, Internet Explorer and Netscape will always open with Mercury Java support active. You can confirm that your Java environment has opened properly by checking the Java console for the following confirmation message: "Loading Mercury Support (version x.x.x.x) (*App*) for QuickTest Professional" (where *App* is IE, NS, SUN, or Oracle).

When you record on a Java applet or application, QuickTest records a Java icon next to the name of the applet or application in the Tree View.

When you record an action on an applet, application or standard Java object, QuickTest records the appropriate object icon next to the step in the Tree View and adds the relevant statement in the Expert view.

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

QuickTest records this step in the Expert View as:

```
Browser("Periodic").Page("Periodic").Applet("periodic.html").JavaCheckBox
    ("Toggle").Set "ON"
```

If you try to record an action on an unsupported or custom Java object, QuickTest records a generic **JavaObject.Click** statement that includes the coordinates of the click and the mouse button (i.e. left or right) that was clicked.

In general, recording on a Java object has the following syntax in the Expert View:

...Applet(*Applet Name***).JavaObject(***Object Name***).Method(***Parameters***)**

You run tests on Java applets and applications just as you would with any other QuickTest test. The Test Results Tree View displays the same icons for Java objects as those used in the Action Tree View and the bottom right pane of the Test Results window displays the applet or application that was captured during the test run, and highlights the Java object for each step in the test results tree.

For more information about running tests, see Chapter 17, "Running Tests."

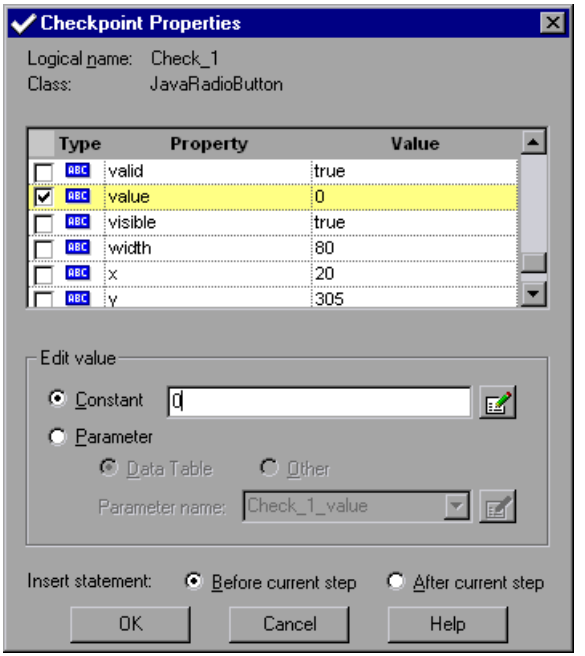
For more information about test results, see Chapter 18, "Analyzing Test Results."

Checking Java Applets, Applications, and Objects

When you create a checkpoint on a Java applet, application or object, QuickTest captures the applet's, application's or object's properties and its values as it captures the property values for any standard Web object.

For example, you can create a checkpoint on a JavaRadioButton object to make sure the radio button is not selected before a step, and another checkpoint to confirm that the same radio button is selected after a step.

The Checkpoint Properties dialog box for the first checkpoint may be displayed as follows:



To create a checkpoint on a Java applet, application or object:

1. In the test tree, highlight the Java applet, application or object you want to check.
2. Right-click the highlighted object in the ActiveScreen and select **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.
3. Select the object for which you want to create a checkpoint and click **OK**. The Checkpoint Properties dialog box opens.
4. Select the properties to check.
5. For each property, select the checkpoint options you want to apply.

   Click **OK**. A tree item with a Java checkpoint icon is added to your test tree.

   For more information on creating checkpoints, see Chapter 6, "Creating Checkpoints."

Retrieving and Setting Java Test Settings

You can configure how QuickTest records and runs tests on a Java applet or application.

You can view or modify commonly used global Java test settings in the Java tab of the Test Settings dialog box.

You can retrieve or set other Java settings during a test run by entering **GetAUTVar** or **SetAUTVar** statements into the test script in the Expert View.

To view or set general Java settings:

1 Choose **Tools** > **Options**. The Options dialog box opens.

2 Click the **Java** tab. The recording settings are displayed.

3 To view additional settings, click **Advanced**.

For a list and description of available settings, see "Java Testing Options," on page 391.

To retrieve or set Java test settings during a test run:

1 In the Expert View, find the location in the script where you want to retrieve or set the test settings.

2 To retrieve a test setting, use the syntax:

NewVar = **JavaUtil.GetAUTVar (** *setting* **)**

To set a test setting, use the format:

JavaUtil.SetAUTVar (*setting*, *new_value* **)**

setting	The name of the setting.
new_value	The value you want to assign to the setting.

For a list and description of the available settings and possible values, refer to the *QuickTest Object Model Reference*.

For example, if you want to set the minimum number of columns for a table to be considered a table object as 2 in an Oracle application, enter:

```
JavaUtil.SetAUTVar "column_number", "2"
```

You could later check the *column_number* setting, by entering:

```
MsgBox JavaUtil.GetAUTVar ("column_number")
```

Working with Objects, Methods, and Events in Your Java Applet or Application

You can create an instance of any Java object (including non-GUI objects) using the **CreateObject** method.

You can view all public methods associated with a Java object in your applet or application as well as the corresponding syntax for each method using the Object Spy.

You can activate any of these methods using the **Object** property. For additional information about the **Object** property, see "Accessing Runtime Object Properties and Methods" on page 374.

The **FireEvent** and **FireEventEx** methods enable you to simulate events on Java objects.

Creating Objects within Your Applet or Application

You can use the **CreateObject** method to create an instance of any Java object within your applet or application.

The **CreateObject** method has the following syntax:

CreateObject(*ClassName*, *[consArg1* **,** *...* **,** *consArgX]* **);**

The *ClassName* argument is the Java class name. *consArg1...ParamX* are the required arguments for that object constructor.

You can activate the methods of an object you create just as you would any returned object from a prior call (without using the **.Object** property).

Activating Methods Associated with a Java Object

You can activate a public Java method for any Java test object using the **Object** property.

If you are not sure which methods your object uses or which arguments you need to send to the method, you can use the Object Spy to view the methods of any object in your applet or application. For more information, see "Viewing Object Methods and Method Syntax Using the Object Spy," on page 33.

Activating the method for a Java object has the following syntax:

JavaObjectName.**Object.***Method_to_activate*( )

For example, suppose the **getBounds** method is supported for your "Enter" JavaButton. To activate the **getBounds** method, you insert the following statement into your test script:

```
Set rect=Browser("Browser").Page("PushButtonDemo").
        Applet("pushbutton.html").JavaButton("Enter").Object.getBounds()
```

If a Java object is returned from a prior **Object** property statement, you can use the returned object in order to activate its methods. You activate the methods of a returned object directly (without using the **Object** property).

For example, you can use the statement below to return the BtnObj.

```
Set BtnObj = Browser("Flight Reservation").Page("Flight Reservation").
    Applet("FlightLogin").JavaButton("OK").Object
```

Then you can invoke the BtnObj's getBounds method and store the returned rectangle, and then invoke the returned rectangle's toString() method.

```
Set Rect = BtnObj.getBounds()
MsgBox Rect.toString()
```

Firing Java Events

You can simulate an event on a Java object during a test run with the **FireEvent** and **FireEventEx** methods. The FireEvent method simulates an event on a Java object using one of several pre-defined event constants (see "Pre-Defined FireEvent Constants," on page 490). If the list of pre-defined constants does not cover the event you want to fire, you can use the FireEventEx function to fire any Java event.

The **FireEvent** method has the following syntax:

JavaObjectName.**FireEvent (** *constant [, EventParam(s)]* **);**

The **FireEventEx** method has the following syntax:

JavaObjectName.**FireEventEx (** *JavaClassName, EventID [, EventParam(s)]* **)**

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

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

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

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

Pre-Defined FireEvent Constants

The FireEvent method includes the following event constants:

micMouseEnter, micMouseExit, micMouseClick, micMousePress, micMouseRelease, micMouseDrag, micMouseMove, micKeyPress, micKeyRelease, micKeyType, micFocusGain, micFocusLost

Adding Java Statements in the Expert View

In addition to the methods described above, QuickTest provides several test objects and methods that you can use with Java applets and applications.

You can record some of these methods while recording on Java applets, applications and objects. You can enter statements with the other methods manually in the Expert View. For more information about programming in the Expert View see Chapter 22, "Testing in the Expert View."

For additional information on these methods, refer to the *QuickTest Object Model Reference*.

32

Testing Multimedia Applications

You can record and run tests on Macromedia Flash, Windows MediaPlayer, and RealPlayer multimedia objects.

Note: Testing on multimedia objects is supported only in QuickTest Professional.

This chapter describes:

- Working with Macromedia Flash Controls
- Working with RealPlayer and Windows MediaPlayer Applications and Controls

About Testing Multimedia Applications

In QuickTest, you can record and run steps on Macromedia Flash controls in Internet Explorer. You can also record and run steps on RealPlayer applications or controls. QuickTest records special multimedia actions on these multimedia objects.

You can create checkpoints on multimedia objects, just as you would for any Web object.

You can also create or enhance tests by entering scripting statements in the Expert View. For information on working in the Expert View, see Chapter 22, “Testing in the Expert View.”

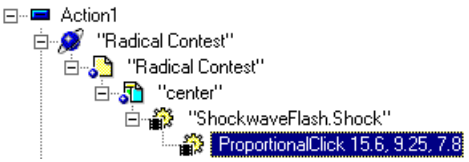
Working with Macromedia Flash Controls

QuickTest supports Macromedia Flash objects that are ActiveX controls in Internet Explorer. For additional information, see Chapter 30, "Testing ActiveX Controls."

Recording and Running Tests on Macromedia Flash Controls

When you record on a Macromedia Flash object, QuickTest records a Flash object icon next to the name of the object in the Tree View and adds the relevant statement in the Expert view.

For example, if you record an action on a FlashButton, the Tree View may be displayed as follows:



QuickTest records this step in the Expert View as:

```
Browser("Radical Contest").Page("Radical
Contest").Frame("center").FlashControl("ShockwaveFlash.Shock").Proportional
Click 15.6,9.25,7.8
```

In general, recording on a Macromedia Flash object has the following syntax in the Expert View:

...FlashControl(*ShockwaveFlash.Shock***).Method(***Parameters***)**

You run tests on Macromedia Flash objects just as you would with any other QuickTest object. The test results tree displays the same icons for Macromedia Flash objects as those used in the test tree. The bottom right pane of the Test Results window displays the Flash object that was captured during the test run.

For more information about running tests, see Chapter 17, "Running Tests."

For more information about test results, see Chapter 18, "Analyzing Test Results."

Checking Macromedia Flash Objects

When you create a checkpoint on a Macromedia Flash object, QuickTest captures the object's properties and its values as it captures the properties and values for any standard Web object.

You can create a checkpoint either while recording or afterward.

For example, you can create a checkpoint on your Flash movie to make sure the movie opens to the right file.

To create a checkpoint on a Macromedia Flash object:

1 In the test tree, highlight the Macromedia Flash object you want to check.

2 Right-click the highlighted object in the ActiveScreen and select **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.

3 Select the object for which you want to create a checkpoint and click **OK**. The Checkpoint Properties dialog box opens.

4 Select the properties you would like to check in the check box column.

5 For each property, select the checkpoint options you want to apply.

Click **OK**. A tree item with a multimedia checkpoint  icon is added to your test tree.

For more information on creating checkpoints, see Chapter 6, "Creating Checkpoints."

Using Scripting Methods in Tests on Macromedia Flash Objects

QuickTest provides several test objects and methods that you can use with MacroMedia Flash objects. You can record some methods while recording on MacroMedia Flash objects. You can also enter methods manually in the Expert View.

For more information about programming in the Expert View, see Chapter 22, "Testing in the Expert View."

For more information about MacroMedia Flash test objects and methods, please refer to the *QuickTest Object Model Reference*.

Working with RealPlayer and Windows MediaPlayer Applications and Controls

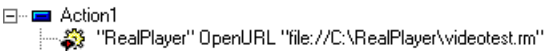
QuickTest supports the RealPlayer and Windows MediaPlayer applications and controls. RealPlayer and MediaPlayer controls are ActiveX controls. For additional information, see Chapter 30, "Testing ActiveX Controls."

Recording and Running Tests on RealPlayer and MediaPlayer Applications and Controls

When you record on a RealPlayer or MediaPlayer application or control, QuickTest records a RealPlayer or MediaPlayer icon next to the name of the object in the Tree View and adds the relevant statement in the Expert view.

Note: In order to record on RealPlayer or MediaPlayer applications, you must open QuickTest before activating the RealPlayer or MediaPlayer application. This includes the RealPlayer StartCenter. Before you open QuickTest to record a RealPlayer application, be sure that the RealPlayer StartCenter icon is not displayed in the status area of the Windows task bar.

For example, if you open a RealPlayer or MediaPlayer clip, the Tree View may be displayed as follows:



QuickTest records this step in the Expert View as:

```
RealPlayer("RealPlayer").OpenURL "file://C:\RealPlayer\videotest.rm
```

If you record a click on a play button in a RealPlayer or MediaPlayer control within a Web browser, the Tree View may be displayed as follows:

QuickTest records this step in the Expert View as:

```
Browser("RealMedia Player").
Page("RealMedia Player").RealControl("video1").Play
```

In general, recording on a RealPlayer or MediaPlayer application has the following syntax in the Expert View:

...RealPlayer (*RealPlayer Application Name* **) . Method (** *Parameters* **)**
...WMPlayer (*RealPlayer Application Name* **) . Method (** *Parameters* **)**

Recording on a RealPlayer or MediaPlayer control within a Web browser has the following syntax in the Expert View:

...RealControl (*Control Name* **) . Method (** *Parameters* **)**
...WMControl (*Control Name* **) . Method (** *Parameters* **)**

You run tests on RealPlayer or MediaPlayer applications or controls just as you would with any other QuickTest test. The test results tree displays the same icons for RealPlayer or MediaPlayer applications or controls as those used in the test tree. The bottom right pane of the Test Results window displays RealPlayer or MediaPlayer as it was captured during the test run.

For more information about running tests, see Chapter 17, “Running Tests.”

For more information about test results, see Chapter 18, “Analyzing Test Results.”

Checking RealPlayer or MediaPlayer Applications or Controls

When you create a checkpoint on a RealPlayer or MediaPlayer application or control, QuickTest captures the object's properties and its values as it captures the properties values for any standard Web object.

You can create a checkpoint either while recording or afterward.

Note: When a clip has not been started, is stopped (not paused) in the middle, or has reached the end, it is unloaded from the control or application and properties checked at that point reflect this status. For example, a checkpoint inserted after a step that stops the clip will show that the URL property is empty, the current position is 0, etc.

In addition to several general object properties, QuickTest captures the following three properties:

Property	Description
State	The current running state of the RealPlayer or MediaPlayer. Possible values are: -1: initial state (no action has been performed on the clip) 0: stopped 1: connecting 2: buffering 3: playing 4: paused 5: seeking
URL	The URL of the open clip.
Current Position	The current time location (in seconds) within the clip.

For example, you can create a checkpoint on your RealPlayer or MediaPlayer application or control to check that your clip stops and returns to the beginning when you click the Stop button.

To create a checkpoint on a RealPlayer or MediaPlayer application or control:

1. In the test tree, highlight the RealPlayer or MediaPlayer application or control you want to check.
2. Right-click the highlighted object in the ActiveScreen and select **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.
3. Select the object for which you want to create a checkpoint and click **OK**. The Checkpoint Properties dialog box opens.
4. Select the properties you would like to check in the check box column.
5. For each property, select the checkpoint options you want to apply.

   Click **OK**. A tree item with a checkpoint  icon is added to your test tree.

   For more information on creating checkpoints, see Chapter 6, “Creating Checkpoints.”

Using Scripting Methods in Tests on RealPlayer or MediaPlayer Applications or Controls

QuickTest provides several test objects and methods that you can use with RealPlayer or MediaPlayer applications or controls. You can record some methods while recording on RealPlayer or MediaPlayer applications or controls. You can also enter methods manually in the Expert View.

For more information about programming in the Expert View, see Chapter 22, “Testing in the Expert View.”

For more information about RealPlayer or MediaPlayer test objects and methods, please refer to the *QuickTest Object Model Reference*.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089 USA

Main Telephone: (408) 822-5200
Sales & Information: (800) TEST-911
Customer Support: (877) TEST-HLP
Fax: (408) 822-5300

Home Page: www.mercuryinteractive.com
Customer Support: support.mercuryinteractive.com