

Astra QuickTest®

User's Guide

Version 5.0



Online Guide




Books
Online


Find

Find
Again


Help



Table of Contents

Welcome to Astra QuickTest.....	12
Using This Guide	12
Online Resources	14
Typographical Conventions	16

PART I: STARTING THE TESTING PROCESS

Chapter 1: Introduction.....	18
Testing with Astra QuickTest.....	19
Astra QuickTest Testing Process	20
Expert View	24
Actions.....	24
Sample Site	24
Managing the Testing Process.....	25



Chapter 2: Astra QuickTest at a Glance	26
Starting Astra QuickTest.....	27
The Astra QuickTest Window	29
Test Pane	31
Display Pane	34
Data Pane.....	34
Debugger Pane	36
Using Astra QuickTest Commands	37

PART II: CREATING TESTS

Chapter 3: Creating Tests	44
About Creating Tests.....	45
Planning a Test.....	46
Recording a Test	47
Creating Checkpoints	51
Understanding Your Test.....	52
Modifying Object Properties in Your Test.....	54
Deleting an Object from the Object Repository	60
Changing the ActiveScreen	62
Managing a Test.....	63



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Chapter 4: Understanding How Astra QuickTest Identifies Objects	65
About How Astra QuickTest Identifies Objects.....	66
Viewing Object Properties Using the GUI Spy	66
Understanding How Astra QuickTest Learns Objects	69
Understanding Dynamic Descriptions of Objects	70
Modifying How Astra QuickTest Identifies Objects.....	71
Chapter 5: Creating Checkpoints.....	74
About Creating Checkpoints.....	75
Checking Objects	76
Adding Checkpoints to a Test.....	77
Understanding the Checkpoint Properties Dialog Box	77
Modifying Checkpoints	80
Chapter 6: Checking Web Objects	81
About Checking Web Objects.....	82
Checking Pages	83
Checking Text	98
Checking Objects	108
Checking Tables.....	115



Chapter 7: Checking Databases.....	121
About Checking Databases	122
Creating a Check on a Database	124
Understanding the Create/Edit Database Checkpoint Dialog Box .	130
Modifying a Database Checkpoint.....	137
Chapter 8: Testing ActiveX Controls.....	138
About Checking ActiveX Controls.....	139
Recording and Running Tests on ActiveX Controls	140
Checking ActiveX Controls	143
Activating an ActiveX Control Method	145
Retrieving and Setting the Values of Properties of ActiveX Controls.....	146
Using Scripting Functions with ActiveX Controls.....	151
Chapter 9: Testing Java Applets.....	153
About Testing on Java Objects.....	154
Recording and Running Tests on Java Applets	155
Checking Java Applets or Objects.....	157
Retrieving and Setting Java Test Settings.....	159
Using Scripting Functions in Tests on Java Applets.....	161

 Books Online
 Find
 Find Again
 Help
 
 Top of Chapter
 Back

Chapter 10: Testing Multimedia Applications.....	172
About Checking Multimedia.....	173
Working with Macromedia Flash Controls.....	174
Working with Real Player Applications and Controls.....	181
Chapter 11: Parameterizing Tests	191
About Parameterizing Tests	192
Setting Parameters as Global or Local.....	197
Parameterizing Steps	198
Parameterizing Checkpoints.....	209
Example of a Parameterized Test	212
Chapter 12: Creating Output Parameters	220
About Creating Output Parameters	221
Creating Page Output Parameters	223
Creating Text Output Parameters.....	228
Creating Object Output Parameters	236
Creating Table Output Parameters.....	242
Chapter 13: Using Regular Expressions	246
About Regular Expressions	247
Using Regular Expressions in Steps	248
Using Regular Expressions in Object Checkpoints	253
Using Regular Expressions in Text Checkpoints.....	257
Regular Expression Syntax	261

Books
Online

Find

Find
Again

Help

Top of
Chapter

Back

Chapter 14: Working with Actions	268
About Working with Actions	269
Using Multiple Actions in a Test	270
Parameterizing Actions	272
Using the Action List	275
Setting Action Properties	278
Creating New Actions	287
Inserting Existing Actions	290
Calling an Action from within an Action	297
Removing Actions From a Test	299
Guidelines for Working with Actions	302
Chapter 15: Working with Data Tables	304
About Working with Data Tables	305
Global and Local Sheets	306
Editing the Data Table	309
Importing Data from a Database	317
Using Formulas in the Data Table	323
Using Data Table Scripting Functions	327

Books
Online

Find

Find
Again

Help

Top of
Chapter

Back

Chapter 16: Handling Unexpected Events and Errors	335
About Handling Unexpected Events and Errors	336
Changing the Status of Exceptions	338
Modifying Exceptions.....	340
Adding New Exceptions.....	342
Deleting Exceptions.....	344

PART III: RUNNING AND DEBUGGING TESTS

Chapter 17: Running Tests	346
About Running Tests	347
Running a Test	348
Using Optional Steps.....	351
Running a Test Batch	355
Chapter 18: Analyzing Test Results	358
About Analyzing Test Results.....	359
The Test Results Window.....	360
Viewing the Results of a Test Run	362
Viewing the Results of a Checkpoint	367
Viewing the Runtime Data Table for a Parameterized Test	369
Printing Test Results	371



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Chapter 19: Debugging Tests.....	372
About Debugging Tests	373
Using the Step Commands.....	374
Pausing Test Runs	375
Setting Breakpoints	376
Deleting Breakpoints	377
Using the Debugger Views	377
Example of Debugging a Test	381

PART IV: ADVANCED FEATURES

Chapter 20: Configuring Event Recording.....	385
About Configuring Event Recording	386
Selecting a Standard Event Recording Configuration	387
Customizing the Event Recording Configuration.....	389
Resetting Standard Event Recording Configuration Settings.....	401
Chapter 21: Enhancing Your Tests with Programming	403
About Enhancing Your Tests with Programming.....	404
Inserting Functions	405
Using Conditional Statements	412
Sending Messages to Your Test Results	417
Adding Comments	419



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Chapter 22: Testing in the Expert View	420
About Testing in the Expert View	421
Understanding the Expert View	422
Programming in the Expert View	428
Enhancing Tests with Comments, Calculations, and Control-Flow Statements.....	441
Chapter 23: Working with Astra QuickTest—for Power Users.....	450
Recording and Running Tests	451
Working with Dynamic Web Content	453
Advanced Web Issues.....	454
Test Maintenance	456

PART V: CONFIGURING ASTRA QUICKTEST

Chapter 24: Setting Astra QuickTest Testing Options.....	458
About Setting Astra QuickTest Options	459
Setting Astra QuickTest Testing Options.....	460
Selecting Astra QuickTest Testing Options	462
Chapter 25: Setting Testing Options for a Single Test	471
About Setting Testing Options for a Single Test.....	472
Setting Testing Options for a Single Test.....	473
Selecting Testing Options for a Single Test	474

Books
Online

Find

Find
Again

Help

Top of
Chapter

Back

Chapter 26: Customizing the Expert View	496
About Customizing Your Test in the Expert View	497
Setting Display Options	499
Personalizing Editing Commands.....	509
Chapter 27: Setting Testing Options from a Test Script.....	512
About Setting Testing Options from a Test Script	513
Setting Testing Options	514
Retrieving Testing Options	517
Controlling the Test Run.....	518
Adding and Removing Run-Time Settings	519

PART VI: WORKING WITH TESTDIRECTOR

Chapter 28: Managing the Testing Process.....	522
About Managing the Testing Process.....	523
Using Astra QuickTest with TestDirector.....	527
Connecting to and Disconnecting from a Project	529
Saving Tests to a Project.....	534
Opening Tests in a Project	537
Running Tests from TestDirector.....	540

Index	543
--------------------	------------



Welcome to Astra QuickTest

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

Using This Guide

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

This guide contains 6 parts:

Part I: Starting the Testing Process

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

Part II: 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 III: 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 IV: 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 applications. **This section is recommended for advanced users of Astra QuickTest.**

Part V: Configuring Astra QuickTest

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

Part VI: Working with TestDirector

Describes how Astra QuickTest interacts with TestDirector, Mercury Interactive's test management tool.



Online Resources



Astra QuickTest includes the following online resources:

Read Me First provides last-minute news and information about Astra QuickTest.

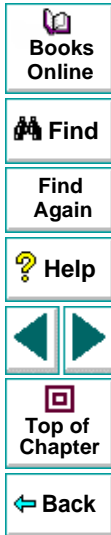
Astra QuickTest Tutorial teaches you basic Astra QuickTest skills and shows you how to start testing your applications.

Books Online displays the *Astra QuickTest User's Guide* in PDF format. Online books can be read and printed using Adobe Acrobat Reader 4.0. Note that in order to view the Books Online you must first install the Acrobat Reader, which is included in the installation package. Check Mercury Interactive's Customer Support Web site for updates to Astra QuickTest online books.

Mercury Tours sample Web site is the basis for many examples in this book. The URL for this Web site is <http://astra.mercuryinteractive.com/mercurytours>.

Astra QuickTest Help describes dialog boxes and toolbar buttons, and provides procedural information.

Astra QuickTest Function Reference gives you online access to the Astra VBScript functions, including a description of each object, a list of the functions (methods) associated with each object, and description syntax and usage of each function (method).



VBScript Reference describes Microsoft's VBScript language, and includes a tutorial and function reference.

Technical Support Online uses your default Web browser to open Mercury Interactive's Customer Support Web site. The URL for this Web site is *<http://support.mercuryinteractive.com>*.

Support Information 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 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**).

Bold

Bold text indicates function names.

Italics

Italic text indicates variable names.

Helvetica

The Helvetica font is used for examples and statements that are to be typed literally.

[]

Square brackets enclose optional parameters.

{ }

Curly brackets indicate that one of the enclosed values must be assigned to the current parameter.

...

In a line of syntax, an ellipsis indicates that more items of the same format may be included. In a program example, an ellipsis is used to indicate lines of a program that were intentionally omitted.

|

A vertical bar indicates that either of the two options separated by the bar should be selected.



Starting the Testing Process

 Books Online

 Find

Find Again

 Help



 Top of Chapter

 Back

Starting the Testing Process

Introduction

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

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

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



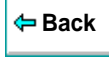
Testing with Astra QuickTest

Astra QuickTest facilitates creating tests on your Web application by recording as you navigate. As you navigate through your site, Astra 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 Astra QuickTest to check the properties of specific objects in your site. For example, you can instruct Astra QuickTest to check that a specific text string appears in a particular location on your Web page, or you can check that a hypertext link goes to the correct URL address.

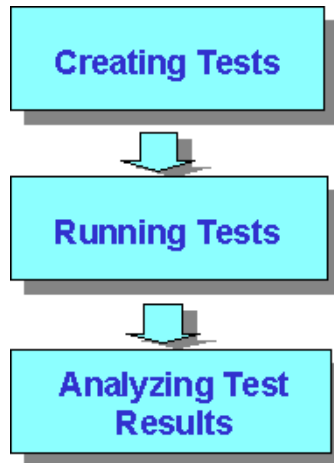
After you record, you can further enhance your test by adding and modifying steps in the test tree. When you run the test, Astra QuickTest connects to your site and performs each step in your test. After you run your test, you can view a report detailing which steps in your test succeeded or failed.

Note that by default, each test includes a single action. You can divide your test into multiple actions. 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](#).



Astra QuickTest Testing Process

Testing with Astra QuickTest involves 3 main stages:



Creating Tests

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

To create a test:

- Record a session on your site.

As you navigate through your site, Astra 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, such as clicking a link or image, or submitting a data form. For more information, see Chapter 3, [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 is functioning correctly. For more information, see Chapter 5, [Creating Checkpoints](#).

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

When you test your site, you can *parameterize* your test to check how your application performs the same operations with multiple sets of data. The data is stored in a table in the Data pane. When you parameterize your test, Astra QuickTest substitutes the parameters in your test with values from the table. During each *iteration* of your test, Astra QuickTest changes the values in the parameterized statements. For more information, see Chapter 11, [Parameterizing Tests](#).

You can also use output parameters to parameterize your test. An *output parameter* is a value retrieved from a parameter in your test during the test run, and entered into your table in the Data pane. You can subsequently use this output parameter 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 12, [Creating Output Parameters](#).



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.

The test runs from the first line in your test and stops at the end of the test. While running, Astra QuickTest connects to your Web site and performs each operation in your test, checking any text strings, objects or tables you specified. If you parameterized your test, Astra 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* 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, [Managing the Testing Process](#).



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](#).

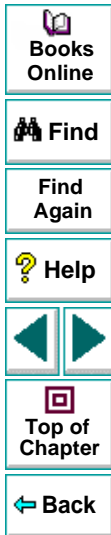
Actions

You can divide your test into sections called “actions.” This enables you to parameterize only part of your test. 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 11, [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://astra.mercuryinteractive.com/mercurytours>.

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

Astra 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 Astra QuickTest, you can create and save tests directly to your TestDirector project. You can also run Astra QuickTest tests from TestDirector and then use TestDirector to review and manage the results. For more information, see Chapter 28, [Managing the Testing Process](#).



Starting the Testing Process

Astra QuickTest at a Glance

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

This chapter describes:

- **Starting Astra QuickTest**
- **The Astra QuickTest Window**
- **Test Pane**
- **Display Pane**
- **Data Pane**
- **Using Astra QuickTest Commands**



Starting Astra QuickTest

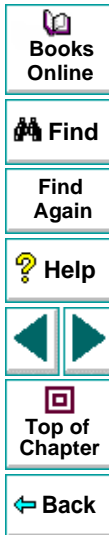


To start Astra QuickTest, click **Programs > Astra QuickTest > Astra QuickTest** (or **Astra QuickTest Professional**) in the Start menu.

The first time you start Astra QuickTest, the Welcome to Astra QuickTest window opens.



You can choose to open the Astra QuickTest tutorial, start recording a new test, open an existing test, or close the welcome window to begin working in a new test.

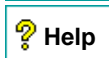


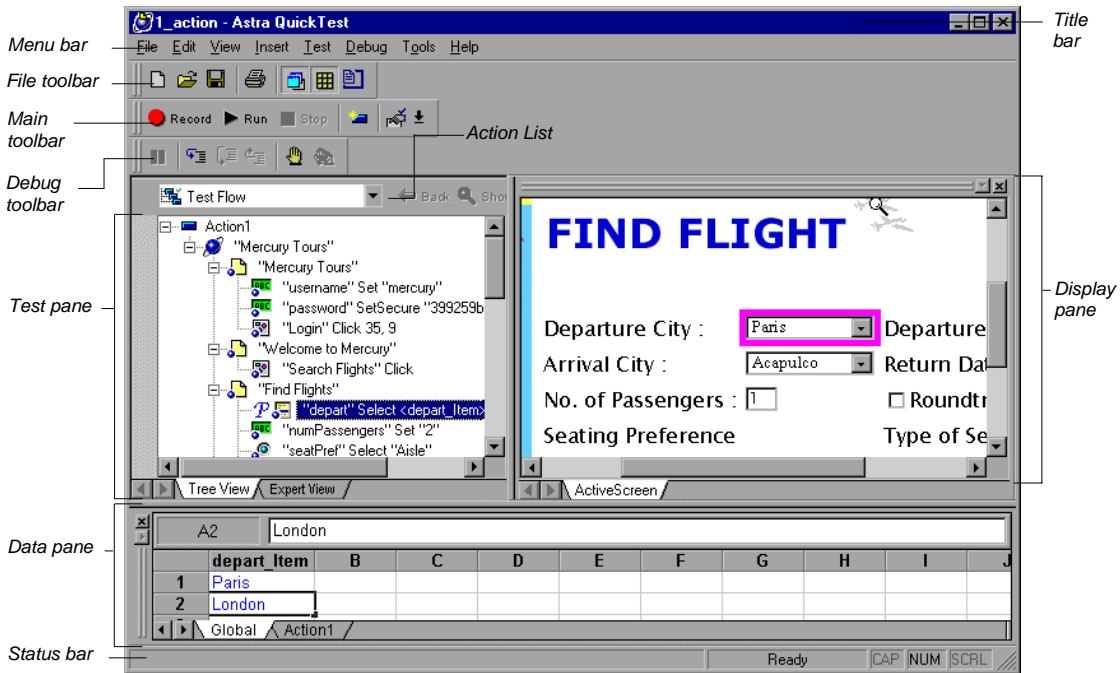
If you do not want this window to appear the next time you start Astra QuickTest, select the **Don't show this dialog next time** check box.


The Astra QuickTest Window

The Astra QuickTest window contains the following key elements:

- *Astra QuickTest title bar*, displaying the name of the currently open test
- *Menu bar*, displaying menus of Astra 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 List*, containing 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. (The Debugger pane can be opened only when a test run pauses at a breakpoint.)
- *Status bar*, displaying the status of the test





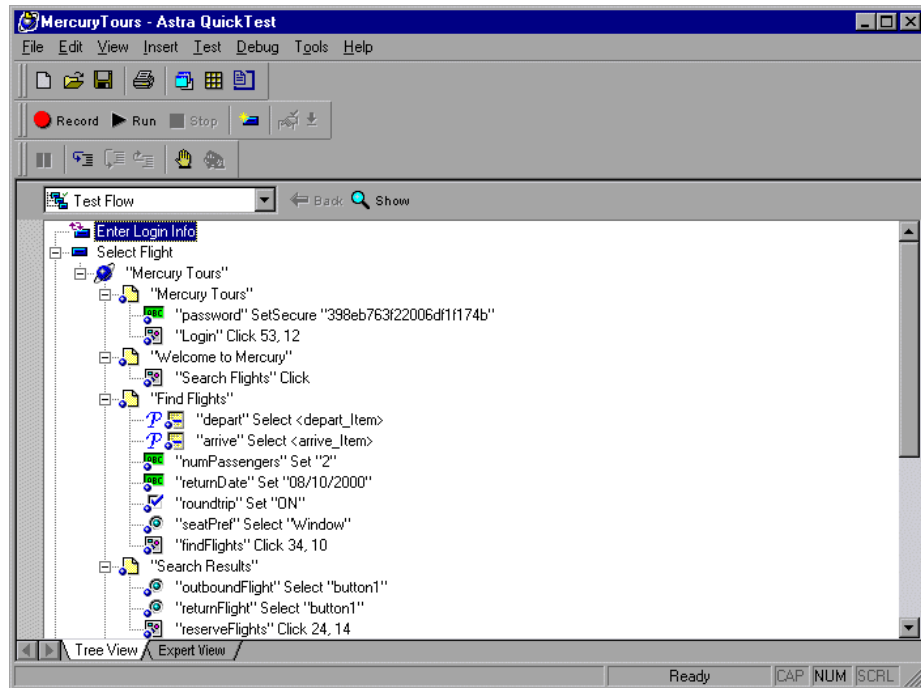
-  Books Online
-  Find
-  Find Again
-  Help
- 
-  Top of Chapter
-  Back

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), Astra QuickTest displays your test in the form of a collapsible icon-based *test tree*. Each operation performed on your Web site is recorded as an icon in your test tree.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

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

Action List

The Action List enables you to view all actions in the test flow or to view the details of a selected action.

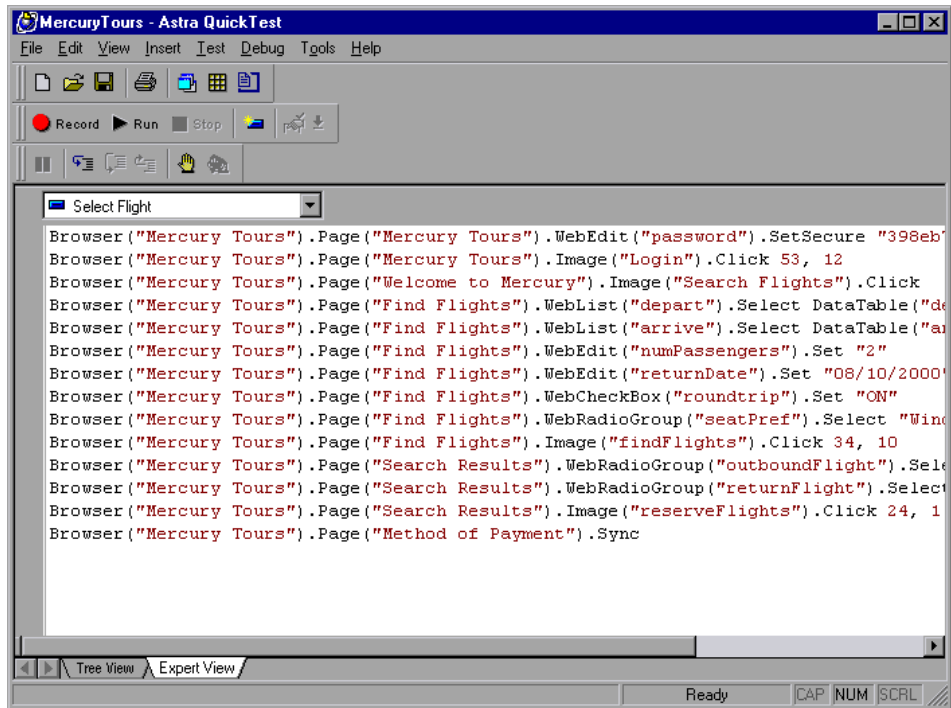
When you have reusable or external actions in your test, the Action List is always visible. If you there are no reusable or external actions in your test, you can choose to show or hide the Action List in the General Tab of the Options dialog box.

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 Action List, or selecting the action in the tree and clicking the Show Action button.



Expert View Tab

In the Expert View tab, Astra 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](#).



Books
Online



Find
Again



Help



Top of
Chapter



Back

Display Pane



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

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, Astra QuickTest inserts the data from the Global tab into the test. For more information, see Chapter 14, [Working with Actions](#).



Books
Online



Find

Find
Again



Help



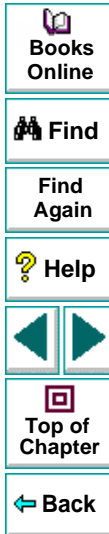
Top of
Chapter



Back

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, Astra 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, choose **View > Debugger Views**. This option is available only when your test run pauses at a breakpoint.

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.

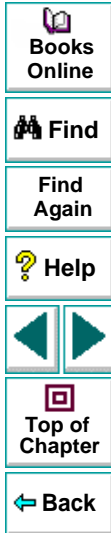
Variables Tab

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, Astra 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 Astra QuickTest Commands

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

Choosing Commands on a Menu

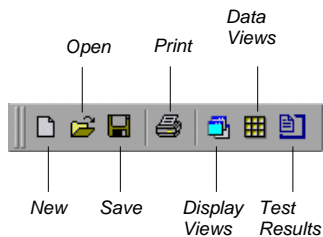
You can choose all Astra QuickTest commands from the menu bar.

Clicking Commands on a Toolbar

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

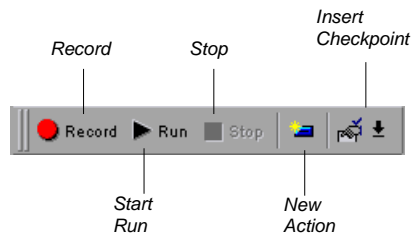
File Toolbar

The File toolbar contains buttons for managing a test. For more information on managing your test, see Chapter 3, [Creating Tests](#). The following buttons appear on the File toolbar:



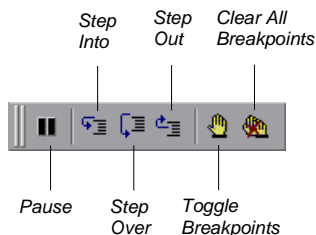
Main Toolbar

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



Books Online

Find

Find Again

Help

Top of Chapter

Back

Executing Commands Using Shortcut Keys

You can execute some Astra QuickTest commands by pressing shortcut keys. The following shortcut keys appear on the corresponding menu commands:

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.
Print	CTRL + P	Prints the active test.
Function Arguments	ALT + ENTER	Opens the Function Arguments dialog box.
Object Repository	CTRL + ENTER	Opens the Repository Editor dialog box for the highlighted test object.
Action Properties	CTRL + ENTER	Opens the Action Properties dialog box for the highlighted action.
Properties	ALT + ENTER	Opens the Script Line Properties dialog box (Expert View only).
Undo	CTRL + Z	Reverses the last command or deletes the last entry you typed (Expert View only).



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Command	Shortcut Key	Function
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	F2	Changes the name of an action or a step (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 function or object, completes the word (Expert View only).



Command	Shortcut Key	Function
Parameter Info	CTRL + SHIFT + SPACE	Displays the syntax of a parameter (Expert View only).
Checkpoint	F12	Creates a checkpoint for an object, or a table.
Output Parameter	CTRL + F12	Creates an output parameter for a text string, an object, or a table.
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.
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 function, the function 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 function, the function is executed in its entirety, but is not displayed in the view.



Books
Online



Find

Find
Again



Help



Top of
Chapter

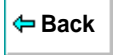


Back

Command	Shortcut Key	Function
Step Out	SHIFT + F11	Runs to the end of the function then pauses execution. (Available only after running a function using Step Into.)
Toggle Breakpoint	F9	Sets or clears a breakpoint in the test.
Clear All Breakpoints	CTRL + SHIFT + F9	Deletes all breakpoints in the test.

[Books Online](#)[Find](#)[Find Again](#)[Help](#)[Top of Chapter](#)[Back](#)

Creating Tests



Creating Tests

Creating Tests

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

This chapter describes:

- **Planning a Test**
- **Recording a Test**
- **Creating Checkpoints**
- **Understanding Your Test**
- **Modifying Object Properties in Your Test**
- **Changing the ActiveScreen**
- **Managing a Test**



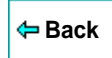
About Creating Tests

Astra QuickTest enables you to generate an automated test by recording the typical processes that you perform on your Web site. As you navigate through your application, Astra 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, for example, clicking a link, or typing data into an edit box.

While recording, you can insert checkpoints into your test. A *checkpoint* compares the current value of the specified property with the expected one in order to help you determine whether or not your Web site is functioning correctly.

When you test your site, you may want to check how it performs the same operations with multiple sets of data. This is called *parameterizing* your test. The data is stored in a table in the Data pane. When you parameterize your test, Astra QuickTest substitutes the parameters in your test with values from the table. During each *iteration*, or *repetition*, of your test, Astra QuickTest inserts a different value in the parameterized statements. Astra QuickTest runs one iteration of your test for each set of values in the Data pane.

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 5, [Creating Checkpoints](#).
- Evaluate the types of events you need to record. If you want to record more or fewer events than Astra 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 11, [Parameterizing Tests](#).
- You can change the way that Astra 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, [Understanding How Astra QuickTest Identifies Objects](#).
- If you are an advanced user, consider using actions to streamline the testing process. For additional information, see Chapter 14, [Working with Actions](#).



Books
Online



Find

Find
Again



Help



Top of
Chapter

← Back

Recording a Test

You create a test by recording the typical processes that users perform. Astra 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, Astra QuickTest records in the standard recording mode. If you are unable to record on an object in a given environment in the standard 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](#) in Chapter 23, [Working with Astra QuickTest—for Power Users](#).



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

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, the browser may prompt you with security alert dialog boxes. You may choose to disable/enable these dialog boxes.
- You can control how Astra QuickTest records and displays your tests by setting testing options in the Options dialog box. For more information, see Chapter 24, [Setting Astra QuickTest Testing Options](#).

To record a test:



- 1 Open Astra QuickTest. For more information, see [Starting Astra QuickTest](#) on page 27.

- 2 Open a test:

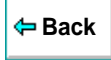


- To create a new test, click the **New** button or choose **File > New**.



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

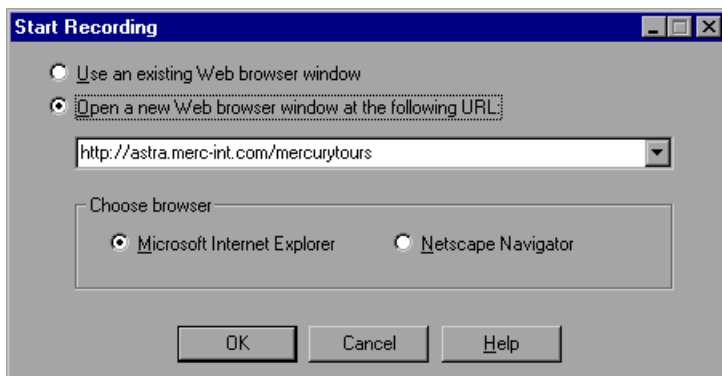
For more information, see [Managing a Test](#) on page 63.





3 Click the **Start Record** button or choose **Test > Record**.

- If this is your first recording session in a test, the Start Recording dialog box opens.



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.

If you select **Open 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 appears as the address.

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

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



Note: The options you select in the Start Recording dialog box also set the Startup settings for the test. For more information about Startup settings, see [StartUp Testing Options](#) on page 475.

- If this is not your first recording session in a test, Astra QuickTest remembers your choices from the previous session. Proceed to step 4.
- 4 Navigate through your Web site. Astra QuickTest records each step you make in the test tree in the Tree View tab.
- 5 You can insert text checkpoints, and object checkpoints, and table checkpoints to compare the current value of the specified property with the expected one, in order to determine whether or not a site is functioning correctly. For more information, see Chapter 5, [Creating Checkpoints](#).
- 6 You can parameterize your test to check how it performs the same operations with multiple sets of data. For more information, see Chapter 11, [Parameterizing Tests](#).
- 7 When you complete your recording session, click the **Stop** button or choose **Test > Stop**.
- 8 To save your test, click the **Save** button or choose **File > Save**. Assign a name to the test. For more information, see [Managing a Test](#) on page 63.



Creating Checkpoints

Astra QuickTest enables you to add checkpoints 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 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, 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 5, [Creating Checkpoints](#).
- **ActiveX controls:** Create checkpoints on ActiveX controls in Microsoft Internet Explorer versions 4.x - 5.01. For more information, see Chapter 8, [Testing ActiveX Controls](#).
- **Java objects:** Create checkpoints on Java applets or objects. All standard Java objects from AWT, JFC, or Oracle toolkits are supported. For more information, see Chapter 9, [Testing Java Applets](#). Note that you can test Java objects only in Astra QuickTest Professional.
- **Multimedia objects:** Create checkpoints on Macromedia Flash objects and for Real Player application and controls. For more information about checking multimedia objects, see Chapter 10, [Testing Multimedia Applications](#). Note that you can test multimedia objects only in Astra QuickTest Professional.

For general information about creating checkpoints, see Chapter 5, [Creating Checkpoints](#).



Find Again



Understanding Your Test

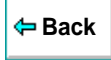
While recording, Astra QuickTest creates a *test tree*—a graphical representation of the navigation you perform on your site. The test tree appears in the Tree View tab. Each step in the tree represents a step performed on your site and browser.





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	<i>Action1</i> is the action name.
"Mercury Tours"	The browser invokes the <i>Mercury Tours</i> site.
"Mercury Tours"	The name of the Web page.



Step	Description
 Checkpoint "Mercury Tours"	A checkpoint that checks statistical information including the load time, the links and the source of images in the <i>Mercury Tours</i> page.
 "username" Set "mercury"	<i>username</i> is the name of the edit box. <i>Set</i> is the method performed on the edit box. <i>mercury</i> is the value of the edit box.
 "password" SetSecure "399259b45"	<i>password</i> is the name of the edit box. <i>SetSecure</i> is an encrypt method performed on the edit box. <i>399259b45</i> is the encrypted value of the password.
 "Login" Click 35, 9	<i>Login</i> is the name of the image. <i>Click</i> is the method performed on the image. <i>52, -8</i> are the x- and y-coordinates where the image was clicked.



Modifying Object Properties in Your Test

As the content of your Web site changes, you can continue to use tests you developed previously. Your test includes all steps you perform, such as clicking hypertext and image links. As Web sites 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 Astra QuickTest can continue to identify it. You can modify the object by modifying one or more of the object's property values in the Object Repository dialog box. For additional information about working with the object repository, see Chapter 4, **Understanding How Astra QuickTest Identifies Objects**.

For example, the Mercury Interactive Web site (*www.mercuryinteractive.com*) has a "Home" hypertext link. Suppose that the text string in this link is changed to "About Mercury Interactive." You need to update your test so that Astra QuickTest will identify the link properly.

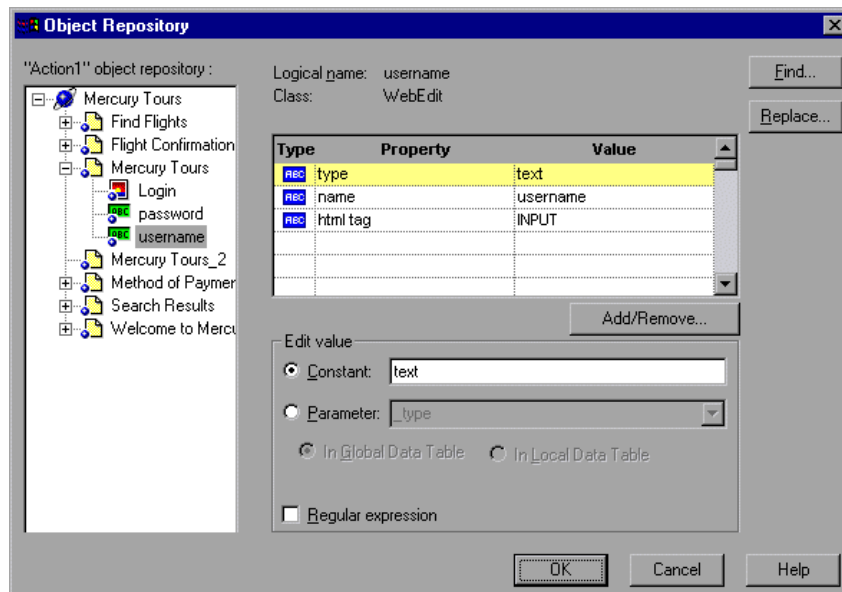
The Object Repository identifies objects on a per action basis. Thus, if the same object appears in several steps within the same action, you only need to modify the object's properties one time. If the object appears again in another action, however, you need to update the object's properties in that action as well. For more information about actions, see Chapter 14, **Working with Actions**.



To modify a step in your test:

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

The Object Repository dialog box opens and displays the properties Astra QuickTest uses to identify the selected object.



Note: If you open the Object Repository from a selected step, the object in the step is selected in the Object Repository tree. If you open the Object Repository from the Tools menu, the first object in the action is selected.

- 2 Select the object you want to modify and highlight the property and value to modify.
- 3 In the **Constant** box, enter a new value for the property.
- 4 Click **OK** to close the dialog box.

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

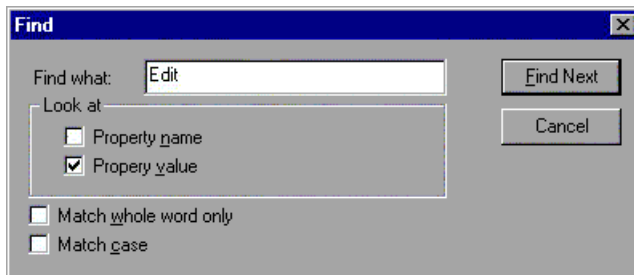
To find a property or value:

- 1 Right-click the step containing an object with the property or value you want to search for, and choose **Object Repository** 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.
- 4 Select **Property name**, **Property value**, or both.
- 5 If you want the search to find only words that exactly match the text you entered, select **Match whole word only**.
- 6 If you want the search to distinguish between upper and lower case letters, select **Match case**.
- 7 Click **Find Next**. The first instance of your search word is displayed.
- 8 To find the next instance, click **Find Next** again.

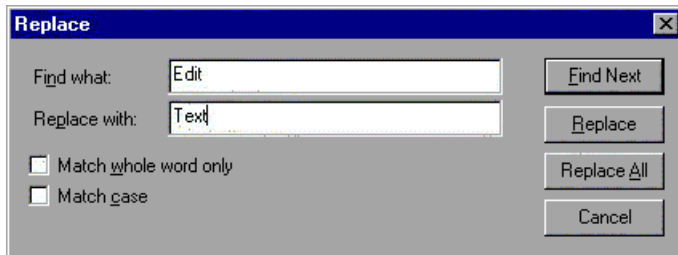
To find and replace a value:

- 1 Right-click the step containing the object with the property or value you want to change, and choose **Object Repository** or choose **Tools > Object Repository**.



The Object Repository dialog box for the selected object opens and displays the properties Astra QuickTest uses to identify the object.

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



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

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

- 4 If you want the search to find only words that exactly match the text you entered, select **Match whole word only**.
- 5 If you want the search to distinguish between upper and lower case letters, select **Match case**.



- 6 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.
- 7 To replace all instances of the word you are searching for with the new value in a single action, click **Replace All**.



Deleting an Object from the Object Repository

When you remove a step from an action your test script, the object still remains in the object repository. If the object in the step you removed does not appear in any other steps within that action, you may want to delete the object from the object repository.

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

To delete an object from the object repository:

- 1 Right-click a step in the action from which you want to delete the object and choose **Object Repository** or choose **Tools > Object Repository**.

The Object Repository dialog box for the selected action opens and displays the properties Astra QuickTest uses to identify the object.

- 2 Right-click the object you want to delete and click **Delete**. A confirmation message appears.
- 3 Click **Yes** to confirm. The object is deleted from the object repository.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Note: Once you confirm that you want to delete the object, you cannot retrieve the object again. Pressing **Cancel** after confirming a delete action will not cancel the delete action.



Changing the ActiveScreen

As the content of your Web site changes, you can continue to use tests you developed previously. You simply change the ActiveScreen display so that Astra 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 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 Astra 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, 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 Astra Test dialog box opens.
- 2 Select a test and click **Open**. The test opens and the title bar displays the test name.



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 Astra 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**. Astra 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.

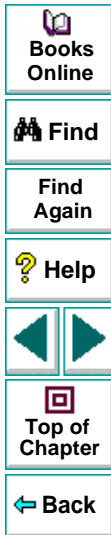
Printing a Test

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.



Creating Tests

Understanding How Astra QuickTest Identifies Objects

This chapter explains how Astra QuickTest identifies objects in your application. It also describes how to modify the way Astra QuickTest identifies an object, which is useful when working with objects that change dynamically.

This chapter describes:

- **Understanding How Astra QuickTest Learns Objects**
- **Understanding Dynamic Descriptions of Objects**
- **Modifying How Astra QuickTest Identifies Objects**



About How Astra QuickTest Identifies Objects

Astra QuickTest identifies each object in your application by its *physical description*: a list of physical properties and their assigned values. An object's physical description always includes the object's logical name and class. The *logical name* is the object's label. The *class* indicates the object's type. Each object belongs to a different class, according to its functionality, for example: Browser, Image, Link (hypertext link), WebList (combo or list box).

Viewing Object Properties Using the GUI Spy

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

To spy on an object:

- 1 Choose **Tools > GUI Spy** to open the GUI Spy dialog box.
- 2 Click the pointing hand. Both Astra QuickTest and the GUI Spy are minimized so that you can point to any object on the open application.



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

The screenshot shows the GUI Spy application window. At the top, it says "Use the pointing hand to click on the object you wish to spy on." Below this is a tree view showing the object hierarchy:

- Browser : Mercury Tours
 - Page : Mercury Tours
 - WebTable : banner_animated
 - WebTable : Sun
 - WebTable : username
 - WebEdit : username** (selected)





Below the tree is a table with two columns: "Properties" and "Values".

Properties	Values
Class Name	WebEdit
value	
name	username
type	text
readonly	0
disabled	0

At the bottom of the window, there is a "Class Name" field and "Close" and "Help" buttons.

Annotations in the image point to:

- object hierarchy tree*: points to the tree view.
- object properties*: points to the table.
- selected property box*: points to the "Class Name" field.

-  Books Online
-  Find
- Find Again
-  Help
- 
-  Top of Chapter
-  Back

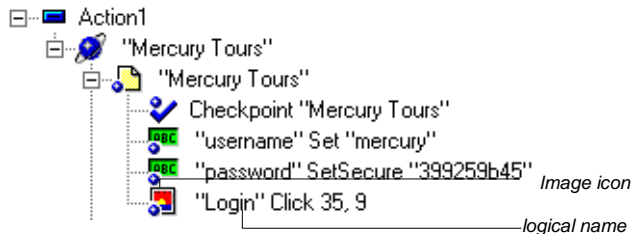
- 4 If you want to view properties for another object within the displayed tree, click the object on the tree.
- 5 If you want to copy an object property or value to the clipboard, click the property or value. The value is displayed in the selected property box. Highlight the text and use CTRL-C to copy the text to the clipboard.



Understanding How Astra QuickTest Learns Objects

Astra 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, Astra QuickTest learns a set of default properties, which it uses to identify objects of that class in your application. For example, by default, Astra QuickTest identifies an image by a physical description that includes the image's HTML tag, the alternate text (if any), and an index (if necessary to make the description unique). The index is a unique number that Astra QuickTest uses to identify an object (in this case the image) on the Web page. The index is numbered in the order in which the object appears in the Web page, starting from top to bottom, and from left to right.



Books Online

Find

Find Again

Help

Top of Chapter

Back

Understanding Dynamic Descriptions of Objects

You can change the properties that Astra 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 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 Astra 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.

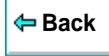
You can modify the default properties for a hypertext link, so that you can identify it by its destination page, rather than by the text in the link. You can use the “href” property to check the destination page instead of using the “text” property to check the link by the text in the link.



Modifying How Astra QuickTest Identifies Objects

Suppose you are testing a Web page that contains an image that is an advertisement. Clicking this image opens the advertiser's Web page. You do not want to identify the image by its name, since the image changes whenever the advertiser changes. Therefore, the value of the name property changes. You would want to create a test that will run no matter what the image's name is. Therefore, you would want Astra QuickTest to identify your image using properties other than the name property.

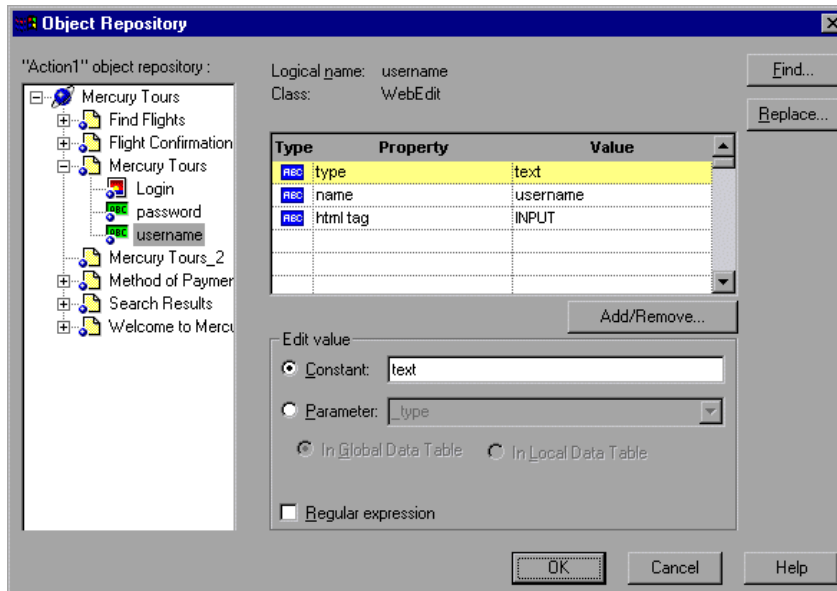
The Object Repository dialog box contains a list of all objects in an action, their properties, and their values. For information about actions, see Chapter 14, **Working with Actions**. By default, Astra QuickTest learns certain properties for each type of object. You can use the Add/Remove Properties dialog box to instruct Astra QuickTest to learn properties of the object other than the default properties.



To modify how Astra QuickTest identifies an object:

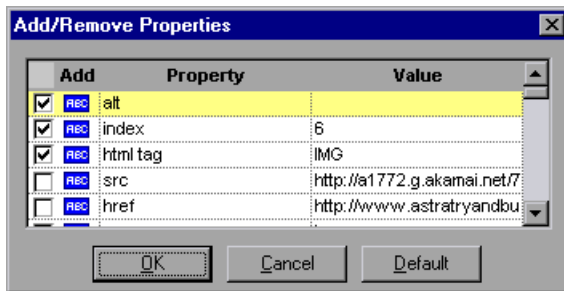
- 1 In the test tree, right-click the object for which you want to modify the identifying properties and choose **Object Repository**.

The Object Repository dialog box opens.



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 Astra QuickTest uses to identify the object. The value for each property is displayed in the Value column.



3 Modify the default properties that Astra 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 dialog box is reactivated.

Tip: After you add a new property, you can modify its value in the Edit Value box in the Object Repository dialog box.



Creating Tests

Creating Checkpoints

You can check objects in your Web site to ensure that it functions as desired.

This chapter describes:

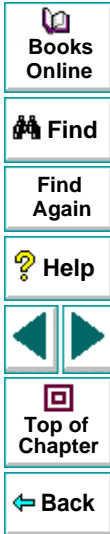
- **Checking Objects**
- **Adding Checkpoints to a Test**
- **Understanding the Checkpoint Properties Dialog Box**
- **Modifying Checkpoints**



About Creating Checkpoints

Astra 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 is functioning correctly.

When you add a checkpoint, Astra QuickTest adds a checkpoint icon under the highlighted step in the test tree. When you run the test, Astra 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 a Web site. You can create checkpoints for 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 6, [Checking Web Objects](#).
- **ActiveX objects:** Perform checks on ActiveX controls in Microsoft Internet Explorer. For more information, see Chapter 8, [Testing ActiveX Controls](#).
- **Java objects:** Perform checks on Java applets or objects. All standard Java objects from AWT, JFC, or Oracle toolkits are supported. For more information, see Chapter 9, [Testing Java Applets](#).
- **Multimedia objects:** Perform checks on Macromedia Flash objects and for Real Player application and controls. For more information about checking multimedia objects, see Chapter 10, [Testing Multimedia Applications](#).

Note: Java and Multimedia objects are supported in Astra 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:



- 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.
- 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

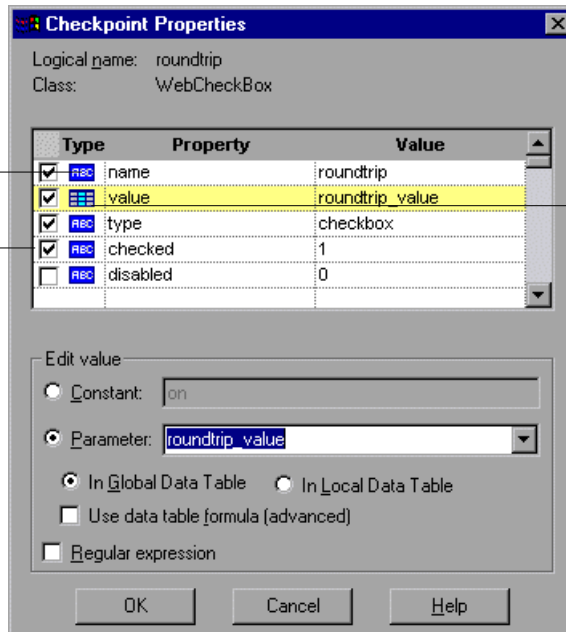
While the Checkpoint Properties dialog box varies slightly depending on the type of object you are checking, the Checkpoint Properties dialog box generally includes the following basic elements:



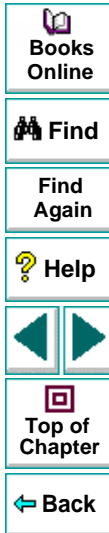
Note: The Text Checkpoint Properties dialog box is quite different from the Checkpoint Properties dialog box described below. For more information see [Understanding the Text Checkpoint Properties Dialog Box](#) on page 100.

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.



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



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, [Understanding How Astra QuickTest Identifies 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 11, [Parameterizing Tests](#).

You can also enter the expected value as regular expression in the **Constant** box or in the data table cells for the parameter. If you choose to enter a regular expression for the expected value, select **Regular expression** at the bottom of the Edit value section.

For more information data tables, see Chapter 15, [Working with Data Tables](#). For more information on regular expressions, see Chapter 13, [Using Regular Expressions](#).



Modifying Checkpoints

If you already created a checkpoint, or if Astra 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 **Function Arguments** or double-click the checkpoint. A checkpoint dialog box opens.
- 2 Modify the properties and click **OK**.



Creating Tests

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:

- **Checking Pages**
- **Checking Text**
- **Checking Objects**
- **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.



Checking Pages

You can check statistical information on 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 Astra QuickTest to create automatic page checkpoints for every page in all tests by selecting the **Add automatic checks for each page** during record check box in the Page Verification 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 Page Verification tab.

You cannot create a new page checkpoint for a page if one has already been created automatically. However, you can modify a page checkpoint that has already been created, as described in [Understanding the Page Checkpoint Properties Dialog Box](#) on page 86.

You can also instruct Astra QuickTest not to perform automatic page checkpoints when you run your test by clearing the **Don't perform automatic checkpoints during test run** check box in the Page Verification tab of the Options dialog box.

For more information, see Chapter 24, [Setting Astra QuickTest Testing Options](#).



Creating a Page Checkpoint

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.

To add a page checkpoint while recording:



- 1 Click the page step in your test tree where you want to add a checkpoint. The ActiveScreen displays the Web page corresponding to the highlighted step.
- 2 Choose **Insert > Checkpoint** 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 86.
- 5 When you are done, 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 and the **ActiveScreen** tab are selected.
- 2 Right-click anywhere on the ActiveScreen and choose **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back



- 3 Select the page item and click **OK**. The Page Checkpoint Properties dialog box opens.
- 4 Specify the settings for the checkpoint in the Page Checkpoint Properties dialog box, as described in [Understanding the Page Checkpoint Properties Dialog Box](#) on page 86.
- 5 When you are done, click **OK** to close the dialog box.

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



Books
Online



Find

Find
Again



Help

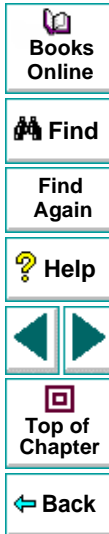
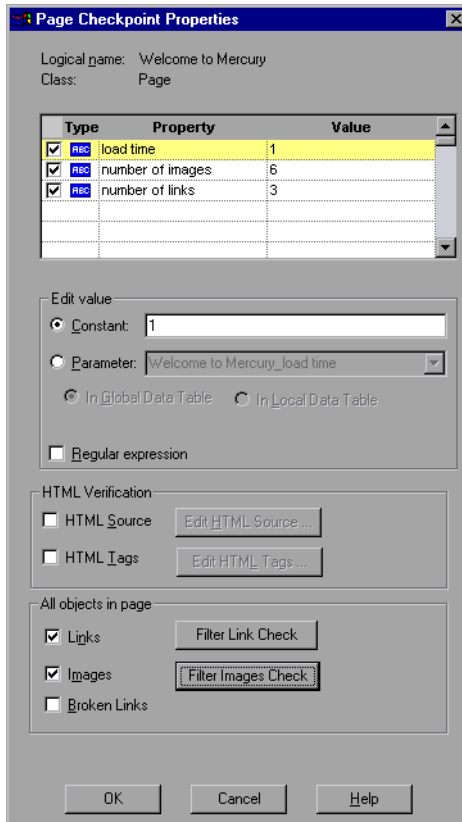


Top of
Chapter

← Back

Understanding the Page Checkpoint Properties Dialog Box

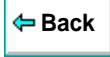
The Page Checkpoint Properties dialog box enables you to choose which properties to check.



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.



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	<p>For each object class, Astra QuickTest recommends default property checks. You can accept the default checks or modify them accordingly.</p> <p>To include a property check, select the corresponding check box.</p> <p>To exclude a property check, clear the corresponding check box.</p>
Type	<p>The  icon indicates that the value of the property is a constant.</p> <p>The  icon indicates that the value of the property is a parameter.</p>
Property	The name of the property to check.
Value	The value of the property to check. 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 Property on page 89.



Editing the Value of a 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 value of the property.
Parameter	Sets the property value as a parameter. For more information, see Chapter 11, Parameterizing Tests .
Global	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Local	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Use data table formula (advanced)	Inserts two columns in the table in the Data pane: the first column contains a formula that checks the validity of output in the second column. Astra QuickTest uses the data in the second (output) column to compute the formula, and inserts a value of TRUE or FALSE in the table cell of the first (formula) column. For more information, see Chapter 15, Working with Data Tables .
Regular expression	Sets the value as a regular expression. For more information, see Chapter 13, Using Regular Expressions .



Checking the HTML Source

In the HTML Source 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 .
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 .



Checking All the Objects in a Page

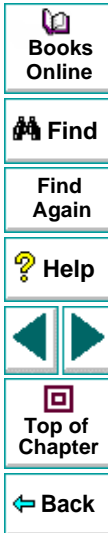
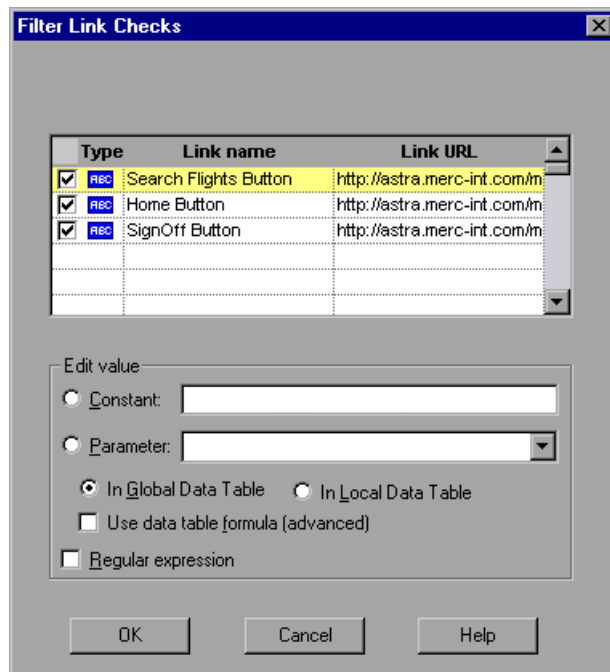
In the All Objects in Page section, you can check all the links, images, and broken links in a page. You can use the following options to check the objects in a page:

Option	Description
Links	Checks the functionality of the links in the page according to your selections in the Filter Link Check dialog box.
Filter Link Check	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 92.
Images	Checks that the images are displayed on the page according to your selections in the Filter Image Check dialog box.
Filter Images Check	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 95.
Broken Links	Checks for broken links in the page.





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 91.



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 a constant. The  icon indicates that the target URL is a parameter.
Link name	The text in the hypertext link.
Link URL	The target URL.



Editing the Value of the Target URL

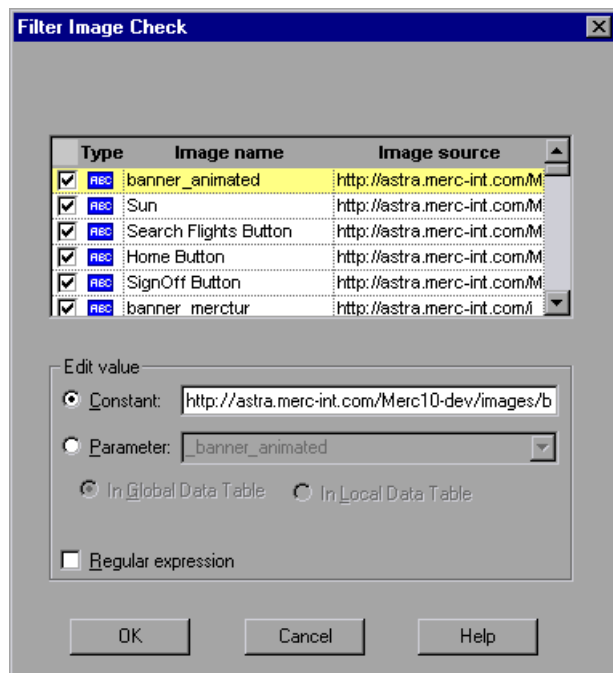
In the Edit Value section, you use the following options to edit the value of the target URL to which the hypertext links:

Option	Description
Constant (default)	Sets the value of the target URL.
Parameter	Sets the target URL as a parameter. For more information, see Chapter 11, Parameterizing Tests .
Global	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Local	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Regular expression	Sets the value as a regular expression. For more information, see Chapter 13, Using Regular Expressions .



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 91.



Books
Online



Find

Find
Again



Help





Top of
Chapter



Back

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	<p>Each image source on the page has a corresponding check box.</p> <p>To check an image source, select the corresponding check box (by default all image sources are selected).</p> <p>To exclude an image source from the page checkpoint, clear the corresponding check box.</p>
Type	<p>The  icon indicates that the image source is a constant.</p> <p>The  icon indicates that the image source is a parameter.</p>
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 value of the path of the image source file.
Parameter	Sets the path of the image source file as a parameter. For more information, see Chapter 11, Parameterizing Tests .
Global	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Local	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Regular expression	Sets the value as a regular expression. For more information, see Chapter 13, Using Regular Expressions .





Checking Text

You can check that a specified text string appears 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 > Text Checkpoint**.
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 100.
- 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 and the **ActiveScreen** tab are 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 100.
- 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 which text to check as well as which text appears before and after the text to check. This is particularly helpful when the text string you want to check appears several times in the same Web page. For example, suppose you want to check that the “Mercury Tours” text string appears in a specific location in the first page of the sample Web site, Mercury Tours. This text string actually appears three times on that Web page.



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.

Text Checkpoint Properties

Check that **Mercury Tours** appears between **Welcome to the** and **website.**

Check for text:

- Constant: Mercury Tours
- Parameter: Mercury_Tours_Check_for_text

In Global Data Table In Local Data Table

Use data table formula (advanced)

Regular expression

Match case Exact match Text not exist

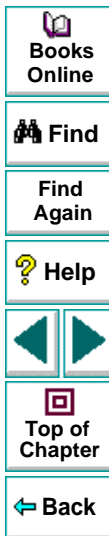
Appears after occurrence 1 of

- Constant: Welcome to the
- Parameter: Mercury_Tours_Appears_after

Appears before occurrence 1 of

- Constant: website.
- Parameter: Mercury_Tours_Appears_before

OK Cancel Help



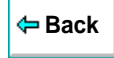
Specifying which Text to Check

In the Check for Text section, you use the following options to specify which text to check:

Option	Description
Constant (default)	The text Astra QuickTest checks when running the test.
Parameter	Sets the text string as a parameter. For more information, see Chapter 11, Parameterizing Tests .
Use data table formula (advanced)	Inserts two columns in the table in the Data pane. The first column contains a formula that checks the validity of output in the second column. Astra QuickTest uses the data in the second (output) column to compute the formula, and inserts a value of TRUE or FALSE in the table cell of the first (formula) column. For more information, see Chapter 15, Working with Data Tables .
In Global Data Table	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
In Local Data Table	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Regular expression	Sets the text string as a regular expression. For more information, see Chapter 13, Using Regular Expressions .



Option	Description
Match case	Conducts a case sensitive search.
Exact match	Checks according to the exact expected text.
Text not exist	Checks that text string does not appear.



Specifying After Which Text the Text to Check Appears

In the Appears After section, you use the following options to specify after which text, if any, the text to check should appear:

Option	Description
<p>Appears after occurrence ___ of (default)</p>	<p>Checks that the text to check appears after the text specified in the Constant or Parameter box.</p> <p>If the identical text string you specify appears more than once on the page, you can specify to which occurrence of the string you are referring.</p> <p>If you accept the default text that Astra QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate.</p> <p>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" appear after the fourth occurrence of the word "the", enter 4 in the Appears after occurrence box.</p> <p>To ignore the text appearing before the text to check, clear this check box.</p>






Option	Description
Constant (default)	Displays the text after which the text to check should appear. Tip: If you modify the text, it is recommended to use a unique string whenever possible so that the occurrence number is always 1.
Parameter	Sets the text string as a parameter. For more information, see Chapter 11, Parameterizing Tests .



Specifying Before Which Text the Text to Check Appears

In the Appears Before section, you use the following options to specify which text, if any, should appear before the text to check:

Option	Description
<p>Appears before occurrence ___ of (default)</p>	<p>Checks that the text to check appears before the specified text in the Constant or Parameter box.</p> <p>Astra QuickTest starts counting occurrences of the Appears before text you specify, from the end of the Appears after string. In other words, it starts looking for the specified text from the text you selected to check.</p> <p>If you accept the default text that Astra QuickTest recommends, the number in the dialog box will be correct. If you modify the recommended text string and the string you specify appears in your highlighted text as well as in the Appears before text, you need to modify the occurrence number accordingly.</p> <p>For example, if you want to check that the words "my hat is the best" appear before the word "hat", enter 2 in the Appears before occurrence box, to show that you want your text to appear before the second occurrence of the word "hat".</p> <p>To ignore the text appearing after the text to check, clear this check box.</p>

 Books Online
 Find
 Find Again
 Help
 
 Top of Chapter
 Back

Option	Description
Constant (default)	Displays the text before which the text to check should appear. Tip: If you modify the text, it is recommended to use a unique string whenever possible so that the occurrence number is always 1.
Parameter	Sets the text string as a parameter. For more information, see Chapter 11, Parameterizing Tests .



Checking Objects

You can check that a specified object appears 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**.

The mouse pointer turns into a pointing hand.

- 2 Click the object 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:

Object	Class
Check box	WebCheckBox
Edit box	WebEdit
Image	Image
Radio button	WebRadio

- 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 111.
- 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 and the **ActiveScreen** tab are 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 object on the ActiveScreen and choose **Insert Checkpoint**.
- 4 Click the object to check. The Object Selection - Checkpoint Properties dialog box opens.
- 5 Select the item you want to check from the displayed object tree. The tree item name depends on the object's class, for example:

Object	Class
Check box	WebCheckBox
Edit box	WebEdit
Image	Image
Radio button	WebRadio

- 6 Click **OK**. The Checkpoint Properties dialog box opens.
- 7 Specify the settings for the checkpoint. For more information, see [Understanding the Checkpoint Properties Dialog Box](#) on page 111.
- 8 Click **OK** to close the dialog box.



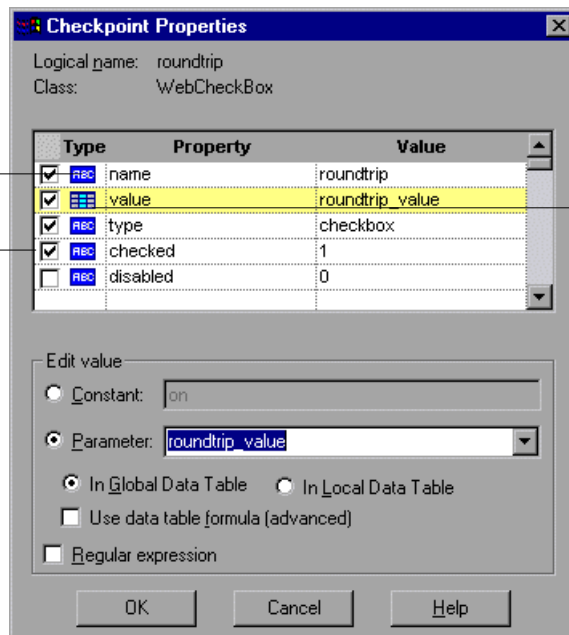
A tree item with a checkpoint  icon is added to your test tree.

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.

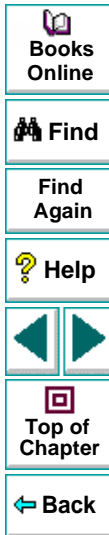
The ABC icon indicates that the value of the property to check is a constant.

The selected checkbox indicates that this property will be checked.



Type	Property	Value
<input checked="" type="checkbox"/> ABC	name	roundtrip
<input checked="" type="checkbox"/> ABC	value	roundtrip_value
<input checked="" type="checkbox"/> ABC	type	checkbox
<input checked="" type="checkbox"/> ABC	checked	1
<input type="checkbox"/> ABC	disabled	0

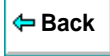
The table icon indicates that the value of the property to check is a parameter.



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 dialog box also displays the default properties of the object you can check in the Properties pane, which lists the properties, their values, and their types:

Pane Element	Description
check box	<p>For each object class, Astra QuickTest recommends default property checks. You can accept the default checks or modify them accordingly.</p> <p>To check a property, select the corresponding check box.</p> <p>To exclude a property check, clear the corresponding check box.</p>
Type	<p>The  icon indicates that the value of the property is a constant.</p> <p>The  icon indicates that the value of the property is a parameter.</p>
Property	The name of the property to check.
Value	<p>The value of the property to check. 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 114.</p>



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 value of the property.
Parameter	Sets the property value as a parameter. For more information, see Chapter 11, Parameterizing Tests .
Global	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Local	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Use data table formula (advanced)	Inserts two columns in the table in the Data pane. The first column contains a formula that checks the validity of output in the second column. Astra QuickTest uses the data in the second (output) column to compute the formula, and inserts a value of TRUE or FALSE in the table cell of the first (formula) column. For more information, see Chapter 15, Working with Data Tables .
Regular expression	Sets the property value as a regular expression. For more information, see Chapter 13, Using Regular Expressions .



Checking Tables

You can check that a specified text string appears 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 Table 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** or click the **Insert Checkpoint** button.

The mouse pointer turns into a pointing hand.

- 2 Click the table. The Object Selection - Checkpoint Properties dialog box opens.



- 3 Select a table item and click **OK**. The Table Checkpoint Properties dialog box opens.

- 4 Specify the settings for the checkpoint. For more information, see [Understanding the Table Checkpoint Properties Dialog Box](#) on page 117.

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

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



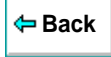
To add a table checkpoint after recording:



- 1 Make sure the **Display Views** button and the **ActiveScreen** tab are 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 and choose **Insert Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.
- 4 Select a table item and click **OK**. The Table Checkpoint Properties dialog box opens.
- 5 Specify the settings for the checkpoint. For more information, see [Understanding the Table Checkpoint Properties Dialog Box](#) on page 117.
- 6 Click **OK** to close the dialog box.



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



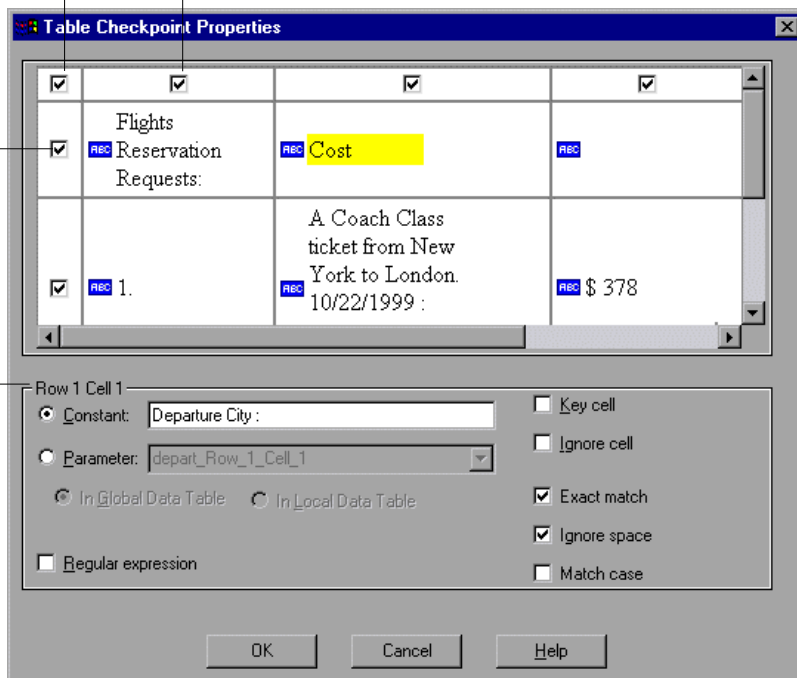
Understanding the Table Checkpoint Properties Dialog Box

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

Select/Clear an entire table Select/Clear an entire column

Select/Clear an entire row

Row and cell indicator



Choosing which Cell to Check

The top part of the Table Checkpoint Properties dialog box displays a grid representing the cells in the table. The grid displays rows and columns of a table. By default the entire table is selected. You can check the entire table, a row, a column, or a cell. Astra QuickTest will only check cells for which the corresponding row and column check boxes are selected.

- To select or clear a row or column, select or clear the corresponding check box.
- To highlight a cell, click it. In the **Row Cell** section, the cell value is displayed in the **Constant** box.



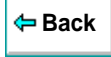
Specifying Cell Contents

The Row __ Cell __ section displays information about the value of the highlighted cell in the table. You can choose from the following tree items:

Option	Description
Row __ Cell __	Displays the row and cell numbers of the highlighted cell in read-only format.
Constant (default)	Displays the value of the text string in the highlighted cell.
Parameter	Sets the value of the cell as a parameter. For more information, see Chapter 11, Parameterizing Tests .
Global	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Local	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .
Regular expression	Sets the value of the cell as a regular expression. For more information, see Chapter 13, Using Regular Expressions .
Key cell	Instructs Astra QuickTest to search for this row according to the contents of this cell.
Ignore cell	Instructs Astra QuickTest not to check the contents of this cell.



Option	Description
Exact match	Checks that the exact text, and no other text, appears in the cell. Clear this box if you want to check that a text string appears in a cell as part of the contents of the cell.
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.
Match case	Conducts a case sensitive search.



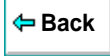
Creating Tests

Checking Databases

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

This chapter describes:

- **Creating a Check on a Database**
- **Understanding the Create/Edit Database Checkpoint Dialog Box**
- **Modifying a Database Checkpoint**



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 Astra QuickTest, you create a database checkpoint based on the results of the query you defined on a database. Astra 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 appears in your test script as a **DbTable.Check CheckPoint** statement.

When you run the test, the database checkpoint compares the current state of the database to the expected data defined in the Edit Database checkpoint 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.

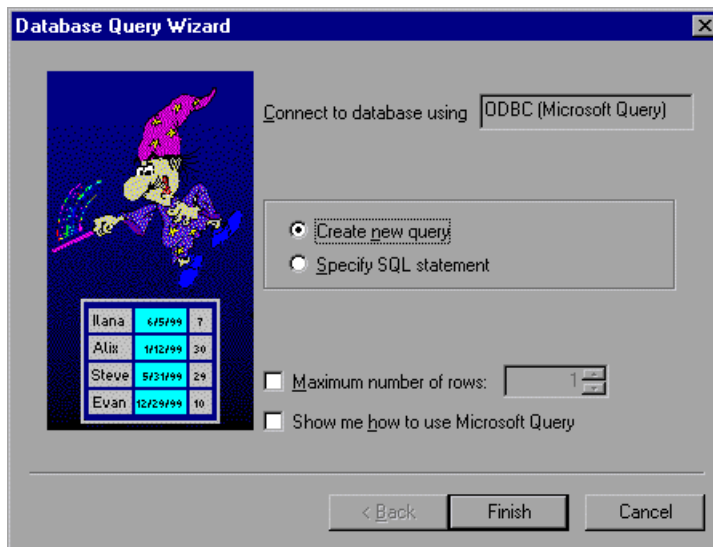
Creating a Database Checkpoint

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:

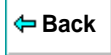
- 1 Choose **Insert > Database Checkpoint**. The Database Query wizard opens.



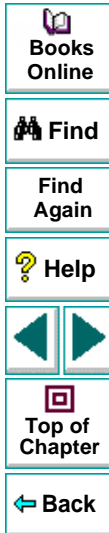
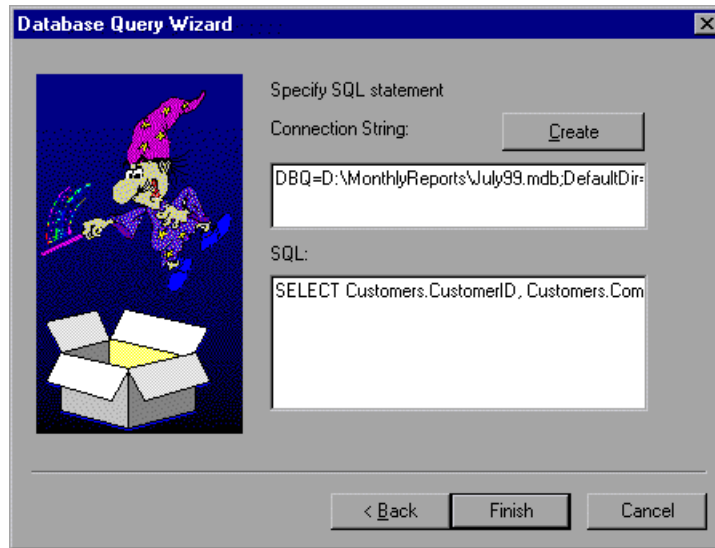
- 2 Select your database selection preferences and click **Next** or **Finish** (depending on which query option you selected). You can choose from the following options:
 - **Create new query:** Opens Microsoft Query, enabling you to create a new query. Once you finish defining your query, you return to Astra QuickTest.



- **Specify SQL statement:** 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. If this check box is cleared, the maximum is 32,000 rows.
 - **Show me how to use Microsoft Query:** Displays an instruction screen before opening Microsoft Query when you click **Finish**. (Enabled only when **Create new query** is selected).
- 3 If you chose **Create new 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 **Finish**.

- **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:** Enter the SQL statement.

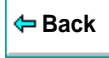
Astra QuickTest takes several seconds to capture the database query and restore the Astra QuickTest window.

- 4 The Create Database Checkpoint dialog box opens. Select the checks to perform on the result set as described in [Understanding the Create/Edit Database Checkpoint Dialog Box](#) on page 130. You can also modify the expected data in the result set.
-  5 Click **OK** to close the Create Database Checkpoint 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. Astra 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:

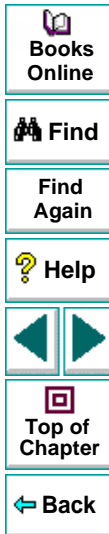
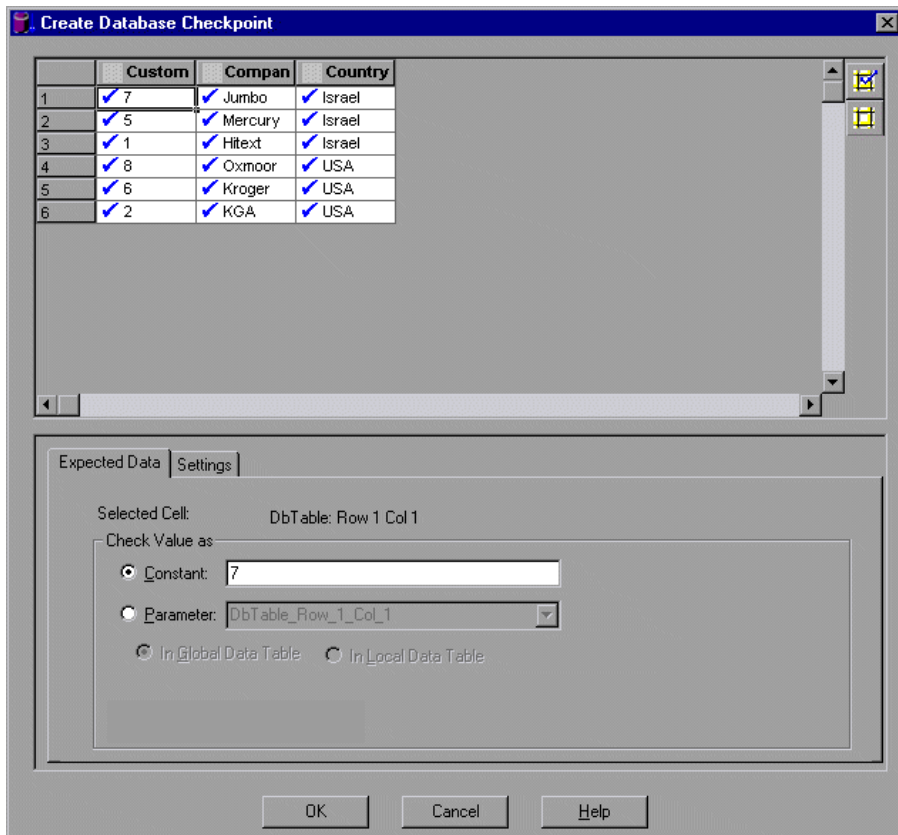
- 1 When Microsoft Query opens during the Insert database checkpoint process, choose a new or an existing data source.
- 2 Define a query.
- 3 When you are done:
 - Version 2.00: Choose **File > Exit and return to Astra QuickTest** to close Microsoft Query and return to Astra QuickTest.
 - Version 8.00 and Version 2000: In the Finish screen of the Query Wizard, select **Exit and return to Astra 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 Astra QuickTest** to close Microsoft Query and return to Astra QuickTest.
- 4 Return to step 4 on [page 128](#) to continue creating your database checkpoint in Astra QuickTest.

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



Understanding the Create/Edit Database Checkpoint Dialog Box

The Create/Edit Database Checkpoint 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.



The dialog box opens as the Create Database Checkpoint dialog box when you are creating the checkpoint and as the Edit Database Checkpoint dialog box when you are editing an existing checkpoint,

Specifying which Cells to Check

The result set pane displays all cells in the captured result set. Cells to be checked are marked with a blue check mark. By default, all cells are marked.

To specify which cells to check:

- 1 Highlight the cells whose check status you want to change.
- 2 Click the **Add Cells to Checkpoint** button to turn on the check for the selected cells, or the **Remove Cells from Checkpoint** button to turn off the check for the selected cells. Alternatively, you can:
 - double-click a cell to change its status
 - double-click a row header to toggle the on/off setting of all the cells in a row
 - double-click a column header to toggle the on/off setting of all the cells in a column
 - double-click the top-left corner to toggle the on/off setting of the entire result set

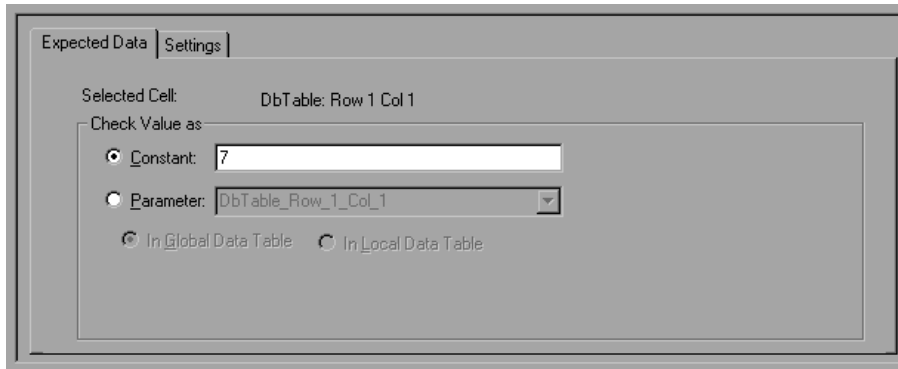



Editing the Expected Data

You can edit the expected database content. The data saved here will be compared to the data retrieved during the test run. You can modify the constant value of the cell and/or parameterize the expected cell content.

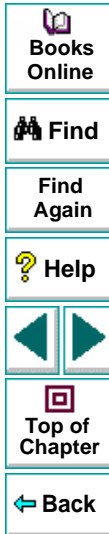
To edit the expected data in the result set:

- 1 Click the **Expected Data** tab.



- 2 Click inside the cell you want to change.
- 3 In the Check Value As box, change the **Constant** value to the desired value or select **Parameter**. The value you enter is displayed in the result set pane. If you select **Parameter**, a special parameter icon  is displayed in the box.

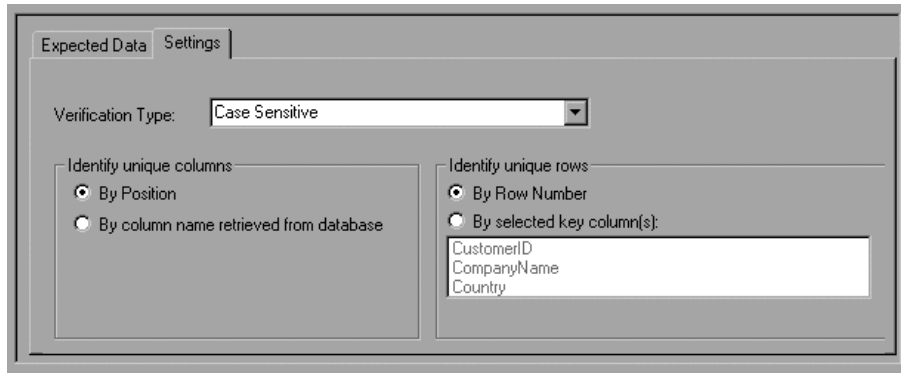
For more information about parameterizing, see Chapter 11, [Parameterizing Tests](#).



- 4 Repeat steps 1 - 3 for each cell you want to modify.
- 5 Click **OK** to save your changes to both tabs and close the Create/Edit Database Checkpoint dialog box.

Setting the Verification Method and Type

In the **Settings** tab, you can specify the verification method and type for the checkpoint.



The default setting is a case sensitive check on the entire result set by column name and row number.



Specifying the Verification Type

Astra QuickTest can verify the contents of a result set in several different ways. The verification type you select applies to the entire result set.

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

- **Case Insensitive:** Astra QuickTest compares the text content of the selected cells including spaces. Only differences in text content between the expected and actual data result in a mismatch.
- **Case Insensitive Ignore Spaces:** Astra QuickTest checks the content in the selected cells according to content, ignoring differences in case and spaces. Astra QuickTest reports only differences in content as a mismatch.
- **Case Sensitive:** (the default): Astra QuickTest compares the text content of the selected cells including spaces. Any difference in case or text content between the expected and actual data results in a mismatch.
- **Case Sensitive Ignore Spaces:** Astra QuickTest checks the data according to case and content, ignoring differences in spaces. Astra QuickTest reports any differences in case or content as a mismatch.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

- **Numeric Content:** Astra QuickTest evaluates the selected data according to numeric values. Astra QuickTest recognizes, for example, that “2” and “2.00” are the same number.
- **Numeric Range:** Astra QuickTest compares the selected data against a numeric range. Both the minimum and maximum values are any real number that you specify. (The **Min** and **Max** fields appear when you select Numeric Range.) This comparison differs from text and numeric content verification in that the actual database data is compared against the range that you define and not against the expected data.

Specifying the Verification Method

You can select the verification method to control how Astra QuickTest identifies columns or rows within a result set. The verification method applies to the entire result set.

Identify Unique Columns:

- **By Position:** Astra QuickTest looks for the selection according to the position, or index, of the columns. For example, a cell in the first column of the database is compared to the same cell in the first column of the result set.
- **By column name retrieved from database:** (default setting): Astra QuickTest looks for the selection according to the column names. For example, a cell in a column named COL_A database is compared to the same cell in the column named COL_A in the result set.



Books
Online



Find

Find
Again




Help



Top of
Chapter

← Back

Identify Unique Rows:

- **By Row Number:** (default setting): Astra QuickTest looks for the selection according to the row number, or index, of the rows. For example, a cell in the first row of the database is compared to the same cell in the first row of the result set.
- **By selected key column(s):** Astra QuickTest looks for the rows in the selection according to the key(s) selected in the **Select key columns** list box, which lists the names of all columns in the result set. If the key selection does not identify a unique row, only the first matching row will be checked. When you select key columns, a key icon  identifies those columns in the result set pane.



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 Repository**.
- 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 **Function Arguments**. The Edit Database Checkpoint dialog box opens.
Modify the settings as described on [page 130](#).



Creating Tests

Testing ActiveX Controls

Astra QuickTest supports testing on ActiveX controls in Microsoft Internet Explorer versions 4.x and higher.

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 Functions with ActiveX Controls**



About Checking ActiveX Controls

Many Web sites include ActiveX controls developed by third-party organizations. Astra QuickTest can run and record tests on these controls as well as check their properties.

Astra 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 5, [Creating Checkpoints](#).

In the Expert View, or using the **Insert > Step** option, you can also activate ActiveX control methods, retrieve and set the values of properties (including run-time 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](#).



Recording and Running Tests on ActiveX Controls

Astra QuickTest records and runs steps on ActiveX controls as it does on any other objects. After running your test, Astra 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](#).



Astra QuickTest records clicks inside the ActiveX control. When you record on an ActiveX control, Astra 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 appear as follows:



Astra QuickTest records this step in the Expert View as:

```
Browser("Untitled").Page("ocxtest.htm").ActiveX("MSACAL.MSACALCtrl.7"
)
.Click 277,116
```

Recording on an ActiveX control has the following syntax in the Expert View:

```
...ActiveX ( ActiveX_control ).Function ( function_parameters )
```



Astra 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, Astra QuickTest records this in the Tree View as follows:



Astra QuickTest records this step in the Expert View as:

```
Browser("Untitled").Page("ocxtest.htm").ActiveX("object").WinCombo  
("ComboBox").Select "May"
```

Note: If an ActiveX control contains another ActiveX control, then Astra 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 Astra 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, Astra 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, Astra 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.

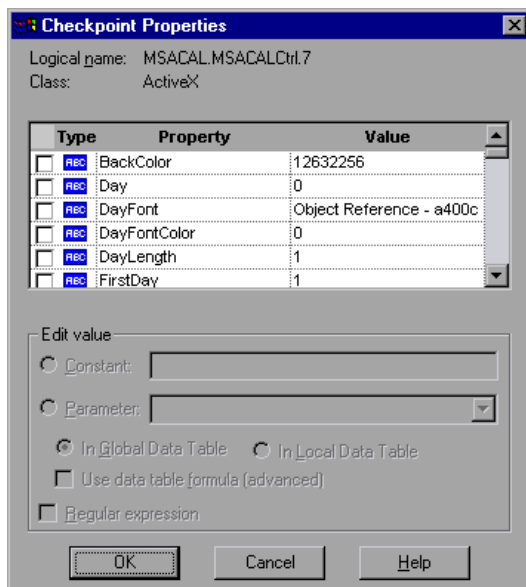
To create a checkpoint on an ActiveX control:


- 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 a checkpoint  icon is added to your test tree. For more information on creating checkpoints, see Chapter 5, [Creating Checkpoints](#).



Activating an ActiveX Control Method

In the Expert View, you can use the **Object** object 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** function (method) is supported for your ActiveX control. To activate the **MakeObjVisible** function, you insert the following statement into your test script:

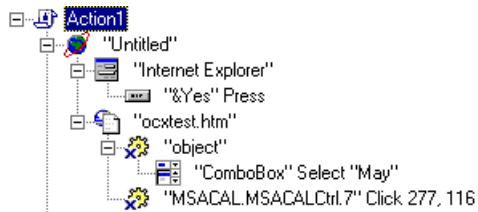
```
Browser("Home").Page("HomePage").ActiveX("MSACAL.MSACALCtrl.7")  
    .Object.MakeObjVisible()
```



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 object.

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").Dialog("Internet Explorer").WinButton("&Yes").Press
Browser("Untitled").Page("ocxtest.htm").ActiveX("object")
    .WinCombo("ComboBox").Select "May"
Browser("Untitled").Page("ocxtest.htm").ActiveX("MSACAL.MSACALCtrl.7"
)
    .Click 277,116
```



Books
Online



Find

Find
Again



Help



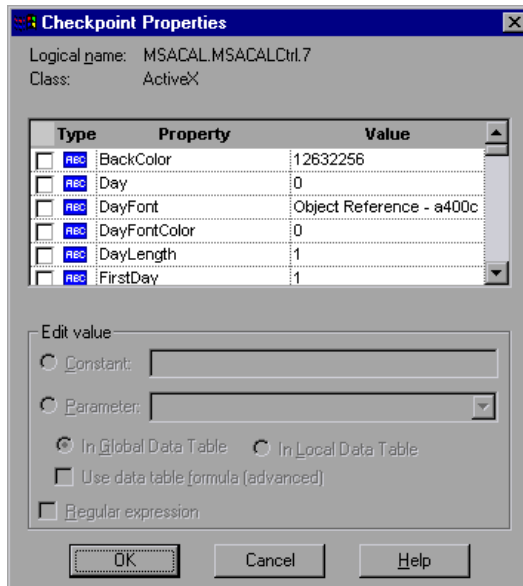
Top of
Chapter



Back

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. 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 object to retrieve and set the day of the month in the calendar, using the Day property.



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("ocxtest.htm").
  ActiveX("MSACAL.MSACALCtrl.7").Object.Day
```

This is displayed in the Tree View as:

```
:: Day =
```

In the Expert View, you can also add a comment to your script and use the **MsgBox** VBScript function to send a message about the retrieved value. Alternatively, you could write this information to the data table or the test report.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

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("ocxtest.htm").ActiveX("MSACAL.MSACALCtrl.7"  
)  
    .Object.Day = 15
```

This is displayed in the Tree View as:

```
;; my_day = Browser("Untitled").Page("ocxtest.htm").ActiveX("MSACAL.MSACALCtrl.7").Object.Day
```



Books
Online



Find

Find
Again



Help

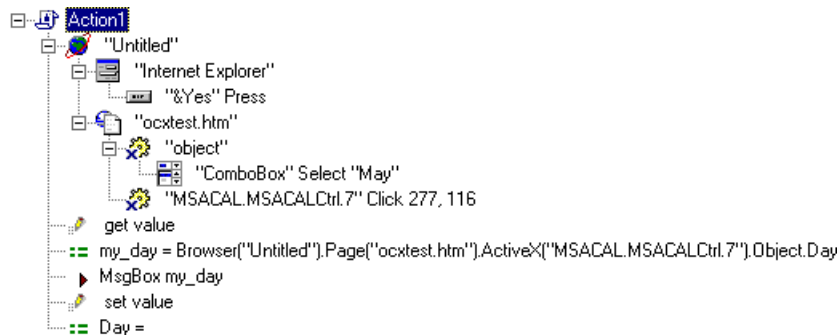



Top of
Chapter



Back

When you are done, your test appears as follows in the Tree View:






**Books
Online**




Find

**Find
Again**



Help





**Top of
Chapter**



Back

Using Scripting Functions with ActiveX Controls

Astra QuickTest provides several scripting functions that you can use with ActiveX controls.

You can record some of these functions while recording on ActiveX controls. You can enter statements manually with the other functions in the Expert View. For more information about programming in the Expert View, see Chapter 22, [Testing in the Expert View](#).

The recordable functions are described below:

Function	Description
Click	Clicks an object
DbClick	Double-clicks an object
Type	Types the specified string



Books
Online

Find

Find
Again

Help



Top of
Chapter

Back

The non-recordable functions are described below:

Function	Description
Exist	Checks that an object exists.
GetProperty	Returns the value of a specified property for an object.
MakeVisible	Scrolls the browser window in order to make the ActiveX control visible.
QueryValue	Returns the current value of a property for an object. The value is taken from the run-time object.
SetProperty	Sets the value of a property in the description of an object.
WaitProperty	Waits for an object property to attain the specified value and then continues the test run. If the specified value is not attained, the function waits until the specified time expires before continuing the test.

For additional information on these functions, refer to the *Astra QuickTest Function Reference*.



Creating Tests

Testing Java Applets

You can record and run tests on Java applets in Microsoft Internet Explorer and Netscape.

Note: Testing on Java objects is supported only in Astra QuickTest Professional.

This chapter describes:

- [Recording and Running Tests on Java Applets](#)
- [Checking Java Applets or Objects](#)
- [Retrieving and Setting Java Test Settings](#)
- [Using Scripting Functions in Tests on Java Applets](#)



About Testing on Java Objects

In Astra QuickTest, you can record and run steps on Java applets in Internet Explorer or Netscape. Astra QuickTest records user actions on applets and on the standard Java objects within the applet. Standard Java objects from the following toolkits are supported:

- AWT from JDK 1.1.x , 1.2.x and 1.3
- Java Foundation Class (JFC) versions 1.0.x and 1.1
- Symantec Visual Café 2.5 and 3.0
- KL Group version 3.x BWT, Field and Table objects
- Oracle Application 10.7 NCA, and Oracle Application 11 NCA (using Forms 4.5.10.x, Forms 5.0, or Forms 6.0), Oracle Developer/2000 1.6.x (Web)

Note: To test Java objects using the Oracle toolkit, you must have the Oracle JInitiator (1.1.5 - 1.1.7.27) installed.

You can create checkpoints on Java applets 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 and objects have Java-specific object names and functions. For information on working in the Expert View, see Chapter 22, [Testing in the Expert View](#).



Recording and Running Tests on Java Applets

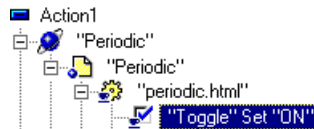
Once you have installed Astra QuickTest, Internet Explorer and Netscape will always open with Mercury Java support active. You can confirm that your Java environment has opened properly by checking the Java console for one of the following confirmation messages: “Mercury Java support is active” or “Init Mercury support”.



When you record on a Java applet, Astra QuickTest records a Java icon next to the name of the applet in the Tree View.

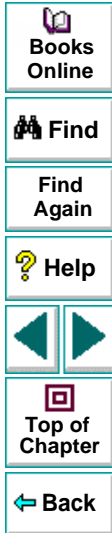
When you record an action on an applet or standard Java object, Astra QuickTest records the appropriate object icon next to the step in the Tree View and adds the relevant statement in the Expert view.

For example, if you record a click on a Java check box, the test tree may appear as follows:



Astra 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, Astra QuickTest records a generic **JavaObject.click** statement that includes the coordinates of the click and the button (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*).
Function(**Parameters**)**

You run tests on Java applets just as you would with any other Astra QuickTest test. The test results tree displays the same icons for Java objects as those used in the test tree and the bottom right pane of the Test Results window displays the applet 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](#).

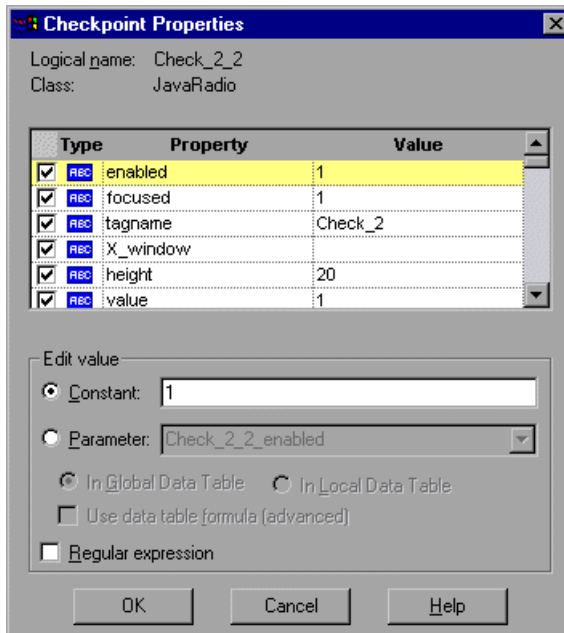
For more information about test results, see Chapter 18, [Analyzing Test Results](#).



Checking Java Applets or Objects

When you create a checkpoint on a Java applet or object, Astra QuickTest captures the applet'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 JavaRadio object to make sure the radio button exists and that it is enabled. The Checkpoint Properties dialog box may appear as follows:



Books
Online

Find

Find
Again


Help



Top of
Chapter

Back

To create a checkpoint on a Java applet or object:

- 1 In the test tree, highlight the Java Applet 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 in the check box column.
- 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 5, [Creating Checkpoints](#).



Retrieving and Setting Java Test Settings

You can configure how Astra QuickTest records and runs tests on a Java applet. You may either modify these settings for the test in general or just for a particular testing session.

You can view or modify the global Java test settings in the Java tab of the Test Settings dialog box. Settings you change in this dialog box apply as long as the settings are maintained.

You can also retrieve or set Java test settings during a test run by entering **GetAUTVar** or **SetAUTVar** statements into the test script in the Expert View. Settings you change using the SetAUTVar statement apply only for the current testing session.

To view or set the test's general Java settings:

- 1 Choose **Test > Settings**. The Test Settings dialog box opens.
- 2 Click the **Java** tab. The available settings and their current values are displayed. If you want to modify a setting proceed to step 3.
- 3 Highlight the setting you want to change.
- 4 Click the value you want to change. A pull-down arrow appears for options with a limited selection. A cursor appears for editable options.
- 5 Enter or select the new value.
- 6 Click **OK**. The changes are saved.



For a list and description of the available settings and possible values, see [Java Testing Options](#) on page 488.

To retrieve or set Java test settings during a test run for the current testing session:

- 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, see [Java Testing Options](#) on page 488.

For example, if you want to set the test to record combo box objects by number rather than item name, enter:

```
JavaUtil.SetAUTVar "record_by_number" , "combo"
```

Later you could return the test to recording all objects by name:

```
JavaUtil.SetAUTVar "record_by_number" , ""
```



Using Scripting Functions in Tests on Java Applets

Astra QuickTest provides several scripting functions that you can use with Java applets.

You can record some of these functions while recording on Java Applets and objects. You can enter statements with the other functions manually in the Expert View. For more information about programming in the Expert View see Chapter 22, [Testing in the Expert View](#).

The recordable functions are described below:

Function	Object(s)	Description
Activate	Applet, JavaInternalFrame, JavaList	Activates the applet, internal frame, or an item in a Java List.
ActivateCell	JavaTable	Activates the specified table cell.
ActivateColumn	JavaTable	Activates the specified table column.
ActivateRow	JavaTable	Activates the specified table row.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back


Function	Object(s)	Description
AddRange	JavaList	selects a range of items and adds them to the current selection.
AddSelection	JavaList	Selects an additional item from the list, such as when a user holds the control button in order to select multiple, non-consecutive items from a list.
Click	Applet, JButton, JavaCheckBox, JavaEdit, JavaInternalFrame, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Clicks the specified location with the specified mouse button.
ClickCell	JavaTable	Clicks the specified cell.
Close	Applet, JavaInternalFrame	Closes the applet or internal frame.
Collapse	JavaList	Collapses an expandable JavaList.



Function	Object(s)	Description
DblClick	Applet, JButton, JavaCheckBox, JavaEdit, JavaInternalFrame, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Double-clicks the specified location with the specified mouse button.
Delete	JavaEdit	Deletes the specified text.
DeselectCell	JavaTable	Deselects the specified cell.
DeselectColumn	JavaTable	Deselects the specified column.
DeselectColumns Range	JavaTable	Deselects the specified range of columns.
DeselectRow	JavaTable	Deselects the specified row.
DeselectRowsRange	JavaTable	Deselects the specified range of rows.
DoubleClickCell	JavaTable	Double-clicks the specified cell.
Drag (for JavaSlider)	JavaSlider	Drags a slider or a scroll bar to a new position.



Function	Object(s)	Description
Drag (for JavaTable)	JavaTable	Performs a mouse drag from the indicated starting cell to the indicated ending cell.
FireEvent	Applet, JButton, JavaCheckBox, JavaEdit, JavaInternalFrame, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Simulates an event on a Java object using one of several pre-defined event constants.
FireEventEx	Applet, JButton, JavaCheckBox, JavaEdit, JavaInternalFrame, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	<p>Simulates an event on a Java object, given the ClassName, Event ID and Event Parameters.</p> <p>Note: You can use the FireEventEx function for any event by specifying its constructors.</p>
Expand	JavaList	Expands an expandable JavaList.

 Books Online
 Find
 Find Again
 Help
 
 Top of Chapter
 Back

Function	Object(s)	Description
ExtendColumn	JavaTable	Adds a column to the specified column.
ExtendColumns Range	JavaTable	Adds a range of columns to the specified column range.
ExtendRow	JavaTable	Adds a row to the specified row.
ExtendRowsRange	JavaTable	Adds a range of rows to the specified row range.
Insert	JavaEdit	Inserts a text string into the specified location in a JavaEdit object.
InvokeMethod	Applet, JButton, JavaCheckBox, JavaEdit, JavaInternalFrame, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Invokes the specified public method of a Java object.
Maximize	Applet, JavaInternalFrame	Maximizes a resizable applet or internal frame.



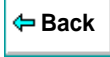
Function	Object(s)	Description
Minimize	Applet, JavaInternalFrame	Minimizes a resizable applet or internal frame.
MouseDown	Applet, JButton, JCheckBox, JEdit, JList, JMenu, JRadio, JSlider, JSpin, JTab, JTable	Uses drag and drop to move an object within the applet or internal frame.
Move	Applet, JavaInternalFrame	Moves an applet or internal frame around the screen.
Press	JButton	Presses a JButton.
Resize	Applet, JavaInternalFrame	Resizes a resizable applet or internal frame to the specified dimensions.
Restore	Applet, JavaInternalFrame	Restores the applet or internal frame to its previous size.
Select (for List)	JList	Selects an item from a Java List.
Select (for Menu)	JMenu	Selects a JMenuItem.
SelectCell	JTable	Selects the specified cell.



Function	Object(s)	Description
SelectCellRange	JavaTable	Selects the specified range of cell.
SelectColumn	JavaTable	Selects the specified column.
SelectColumn Range	JavaTable	Selects the specified range of columns.
SelectItem	JavaTab	Selects the specified tab panel.
SelectRange	JavaList	Selects the indicated range of items from a list.
SelectRow	JavaTable	Selects the specified row.
SelectRowsRange	JavaTable	Selects the specified range of rows.
Set	JavaEdit, JavaCheckBox, JavaRadio, JavaSpin	Sets the value or state of the Java object.
SetCellData	JavaTable	Sets the cell contents with the specified data.
SetInsertPos	JavaEdit	Places the cursor in the specified location in a JavaEdit object.



Function	Object(s)	Description
UnSelect	JavaList	Deselects an item from a Java List.
UnSelectRange	JavaList	Deselects the indicated range of items from a list.

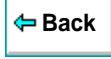


The non-recordable functions are described below:

Function	Object(s)	Description
Exist	Applet, JButton, JavaCheckBox, JavaEdit, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Checks that an object exists.
GetProperty	Applet, JButton, JavaCheckBox, JavaEdit, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Returns the specified value of a property for an item. The value is taken from the object repository.
GetAUTVar	JavaUtil	Retrieves the current value for the specified Java test setting.



Function	Object(s)	Description
MethodWizard	Applet, JButton, JavaCheckBox, JavaEdit, JavaInternalFrame, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Activates the MethodWizard for the object. The MethodWizard enables you to view the methods associated with the selected object. You can invoke any of the public methods displayed for the object using the InvokeMethod function.



Function	Object(s)	Description
QueryValue	Applet, JButton, JavaCheckBox, JavaEdit, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Returns the current value of a property for an item. The value is taken from the runtime object.
SetProperty	Applet, JButton, JavaCheckBox, JavaEdit, JavaList, JavaMenu, JavaObject, JavaRadio, JavaSlider, JavaSpin, JavaTab, JavaTable	Sets the specified value of a property for an item.

For additional information on these functions, refer to the *Astra QuickTest Function Reference*.



Creating Tests

Testing Multimedia Applications

You can record and run tests on Macromedia Flash and Real Player Multimedia objects.

Note: Testing on Multimedia objects is supported only in Astra QuickTest Professional.

This chapter describes:

- **Working with Macromedia Flash Controls**
- **Working with Real Player Applications and Controls**

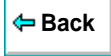


About Checking Multimedia

In Astra QuickTest, you can record and run steps on Macromedia Flash in Internet Explorer. You can also record and run steps on Real Player applications and controls. Astra QuickTest records clicks and other 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

Astra QuickTest supports Macromedia Flash objects that are ActiveX controls in Internet Explorer. For additional information, see Chapter 8, [Testing ActiveX Controls](#).



Recording and Running Tests on Macromedia Flash Controls

When you record on a Macromedia Flash object, Astra 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 a click on a Flash object, the Tree View may appear as follows:



Astra QuickTest records this step in the Expert View as:

```

Browser("Radical Contest").Page("Radical
Contest").Frame("center").FlashControl("ShockwaveFlash.Shock").Proportio
nalClick 15.6,9.25,7.8
  
```



In general, recording on a Macromedia Flash object has the following syntax in the Expert View:

...FlashControl(*Movie Name*).Function(Parameters)

You run tests on Macromedia Flash objects just as you would with any other Astra 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](#).

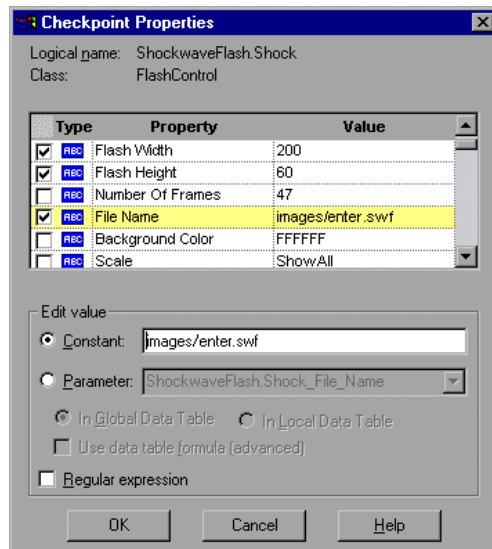


Checking Macromedia Flash Objects



When you create a checkpoint on a Macromedia Flash object, Astra 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. The Checkpoint Properties dialog box might be displayed as follows:



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 checkpoint  icon is added to your test tree.

For more information on creating checkpoints, see Chapter 5, [Creating Checkpoints](#).



Using Scripting Functions in Tests on Macromedia Flash Objects

Astra QuickTest provides several scripting functions that you can use with Macromedia Flash objects.

You can record some of these functions while recording on Macromedia Flash objects. You can enter the other functions manually in the Expert View. For more information about programming in the Expert View see Chapter 22, [Testing in the Expert View](#).

The recordable functions are described below:

Function	Description
ProportionalClick	Moves the mouse to the specified relative location proportional to the size of the entire screen, waits the specified time and then clicks.
Type	Types the specified string.



The non-recordable functions are described below:

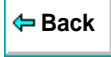
Function	Description
ClipGotoFrame	Goes to the specified clip frame according to the frame number.
ClipGotoLabel	Goes to the specified clip frame according to the frame label.
ClipPlay	Begins play on a movie clip.
ClipStop	Stops playing a movie clip.
GetClipCurrentFrame	Retrieves the current frame number of a movie clip.
GetClipCurrentLabel	Retrieves the current frame label of a movie clip.
GetPropertyByName	Retrieves the value of the specified ActiveX property.
GetVariable	Retrieves the value of the specified movie variable.
GotoFrame	Jumps to the specified frame number.
MakeVisible	Scrolls the control into view.
Play	Plays a movie.
ProportionalMove	Moves the mouse on the Flash object to the specified relative location proportional to the size of the entire screen, and then waits the specified wait time.



Function	Description
SetVariable	Sets the value of the specified movie variable.
Stop	Stops a movie.

For more information on these functions, please refer to the *Astra QuickTest Function Reference*.

Note: You can also access the internal methods (functions) of the ActiveX control used by the Flash control, using the `.Object` object. For more information about accessing internal methods see Chapter 8, [Testing ActiveX Controls](#).



Working with Real Player Applications and Controls

Astra QuickTest supports the Real Player application and Real Player controls. Real Player controls are ActiveX controls. For additional information, see Chapter 8, [Testing ActiveX Controls](#).

Recording and Running Tests on Real Player Applications and Controls



When you record on a Real Player application or control, Astra QuickTest records a Real Player 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 RealPlayer applications, you must open Astra QuickTest before activating the RealPlayer application. This includes the RealPlayer StartCenter. Before you open Astra QuickTest to record a RealPlayer application, be sure that the RealPlayer StartCenter icon does not appear in the status area of the Windows task bar.

For example, if you record a click on the play button in Real Player, the Tree View may appear as follows:

```

Action1
├── "RealPlayer" OpenURL "file://C:\RealPlayer\wideotest.rm"

```



Astra QuickTest records this step in the Expert View as:

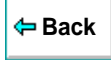
```
RealPlayer("RealPlayer").OpenURL "file://C:\RealPlayer\videotest.rm"
```

If you record a click on a play button in a Real Player control within a web browser, the Tree View may appear as follows:



Astra QuickTest records this step in the Expert View as:

```
Browser("RealMedia Player").
Page("RealMedia Player").RealControl("video1").Play
```



In general, recording on a Real Player application has the following syntax in the Expert View:

...RealPlayer (*RealPlayer Application Name*) . Function (Parameters)

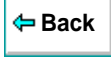
Recording on a Real Player control within a Web browser has the following syntax in the Expert View:

...RealControl (*Control Name*) . Function (Parameters)

You run tests on Real Player applications or controls just as you would with any other Astra QuickTesttest. The test results tree displays the same icons for Real Player applications or controls as those used in the test tree. The bottom right pane of the Test Results window displays Real Player 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](#).

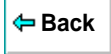


Checking Real Player Applications or Controls

When you create a checkpoint on a Real Player application or control, Astra 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.

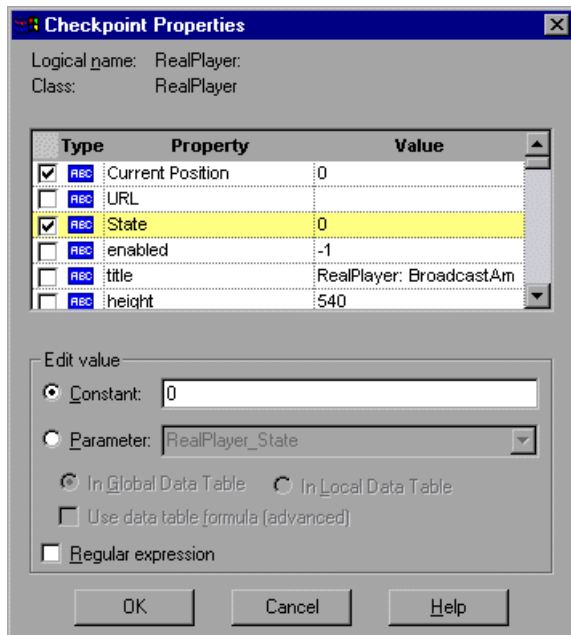


In addition to several general object properties, Astra QuickTest captures the following three properties:


Property	Description
State	<p>The current running state of the real player. Possible values are:</p> <ul style="list-style-type: none"> -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 Real Player application or control to check that your clip stops and returns to the beginning when you click the Stop button. The Checkpoint Properties dialog box may appear as follows:



To create a checkpoint on a Real Player application or control:

- 1 In the test tree, highlight the Real Player 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 5, **Creating Checkpoints**.




Using Scripting Functions in Tests on Real Player Applications or Controls

Astra QuickTest provides several scripting functions that you can use with Real Player applications or controls.

You can record these functions while recording on Real Player applications or controls. You can also enter the functions manually in the Expert View. For more information about programming in the Expert View, see Chapter 22, [Testing in the Expert View](#).

The recordable functions are described below:

Function	Description
Close	Closes the RealPlayer application. (Applies to RealPlayer applications only).
OpenUrl	Opens the URL containing the Real clip. (Applies to RealPlayer applications only).
Pause	Pauses playing at the current time position.
Play	Plays the clip from the current time position.
ProportionalClick	Clicks on the Real Control or RealPlayer based on the relative location of the click proportional to the size of the entire screen.

 Books Online
 Find
 Find Again
 Help
 
 Top of Chapter
 Back

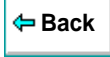
Function	Description
Seek	Jumps to the specified time spot in the clip.
Stop	Stops playing the clip and removes it from memory.
WaitClipEnd	Waits for the end of the clip.
WaitClipTime	Waits for the clip to reach the specified time spot.

The non-recordable functions are described below:

Function	Description
GetCurrentTime	Retrieves the current time position of a Real object.
GetPropertyByName	Retrieves the value of the specified ActiveX property. (Applies to RealControls only).
MakeVisible	Scrolls the control into view. (Applies to RealControls only).



Note: Astra QuickTest automatically inserts a **WaitClipTime** step before the **Pause** and **Stop** steps, and inserts a **WaitClipEnd** step when the clip reaches the end, in order to assure that the following step occurs at the appropriate time position.



Creating Tests

Parameterizing Tests

Astra 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:

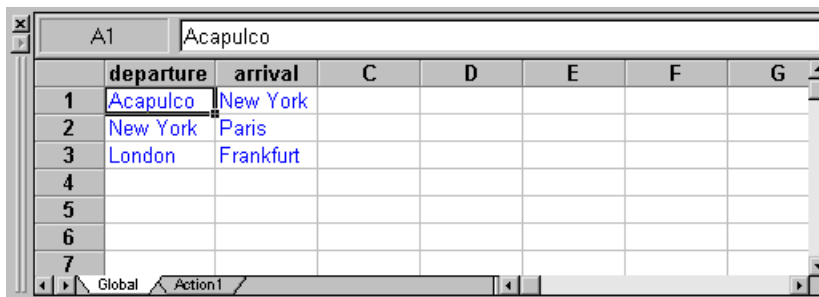
- **Parameterizing Steps**
- **Setting Parameters as Global or Local**
- **Parameterizing Checkpoints**
- **Example of a Parameterized Test**



About Parameterizing Tests

You can use Astra 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.

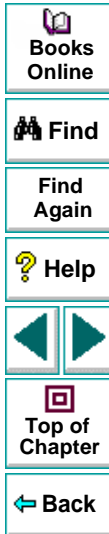
You start by recording a test that performs a set of actions. After you finish recording, you can parameterize certain constants in the test so that it will run the same set of actions many times. In each repetition, or *iteration*, Astra QuickTest substitutes the constant value with a parameter value. You supply the list of possible values for a parameter in a table in the Data pane.



The screenshot shows a window titled 'A1' with a text field containing 'Acapulco'. Below it is a table with columns labeled 'departure', 'arrival', 'C', 'D', 'E', 'F', and 'G'. The first three rows contain data for the 'departure' and 'arrival' columns, representing different city pairs. The 'C' through 'G' columns are empty. The table has a scrollable interface with a vertical scrollbar on the right and a horizontal scrollbar at the bottom. The status bar at the bottom indicates 'Global' and 'Action1'.

	departure	arrival	C	D	E	F	G
1	Acapulco	New York					
2	New York	Paris					
3	London	Frankfurt					
4							
5							
6							
7							

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 Astra QuickTest submits for all the parameters during a single iteration of the test. When you run your test, Astra 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.

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.



- flights
- home
- sign off

FIND FLIGHT

Departure City :

Arrival City :

No. of Passengers :

Seating Preference

Aisle

Window

None

Departure Date :

Return Date :





Roundtrip ticket

Type of Seat

First

Business

Coach

-  Books Online
-  Find
- Find Again
-  Help
- 
-  Top of Chapter
-  Back

The site returns the available flights for the requested itinerary.



search results
FLIGHTS

Flight departing from Acapulco to New York on
04/05/2000

Flight	Departure time	Cost
<input checked="" type="radio"/> Blue Sky Air 030	8am	\$ 658
<input type="radio"/> Blue Sky Air 031	1pm	\$ 587
<input type="radio"/> Blue Sky Air 032	5pm	\$ 622
<input type="radio"/> Blue Sky Air 033	11pm	\$ 539

[continue...](#) [start over](#)

Books Online

Find

Find Again

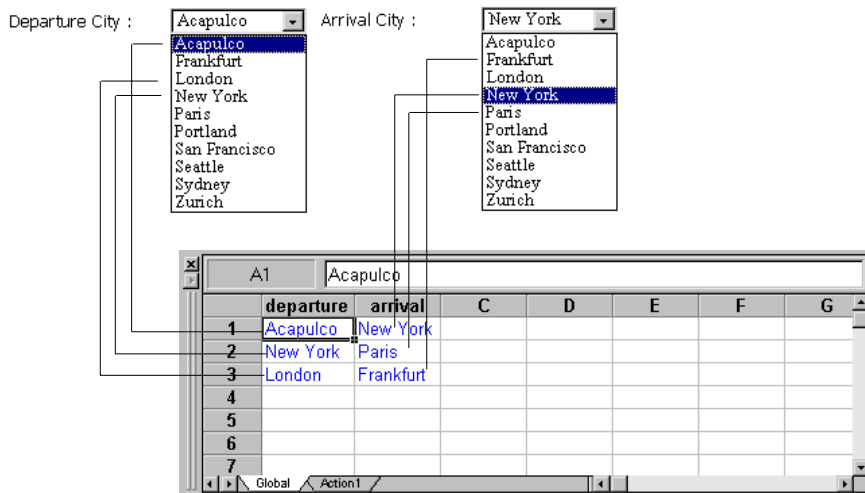
Help

Top of Chapter

Back

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 parameter, and add multiple sets of data, one for each itinerary, into the table linked to the test.

When you run the test, Astra QuickTest submits a separate query for each itinerary.



When you add parameters to your test, you can parameterize a step recorded in your test or a checkpoint added to your test. When you parameterize a step, you parameterize either the *object* that you navigate in your Web page or the *method* by which you navigate. 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 with multiple sets.



You can also parameterize your test by creating output parameters, which retrieve variables from the test while it runs and insert them into a table in the Data pane so that you can use them as input later in the test. For more information, see Chapter 12, [Creating Output Parameters](#).

Note: After running, you can view the results of the values used in a parameterized test in the Runtime Data table. For more information, see [Viewing the Runtime Data Table for a Parameterized Test](#) on page 369.



Setting Parameters as Global or Local

When you parameterize a step, you must decide whether you want to make it a *global parameter* or a *local parameter*.

Global parameters take data from the global tab in the data table. The global tab 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 Testing Options](#) on page 477.

Local parameters take data from the action's local tab in the data table. The data in the action's local tab, replaces local 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 local sheet, or to run iterations on specified rows within the action's local sheet.

For more information about setting local iteration preferences, see [Setting the Run Properties for an Action](#) on page 284.



Parameterizing Steps

You can parameterize a step while recording your test or afterward. You parameterize a step in your test tree. 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 are actually parameterizing either the object or the method. For an example of a parameterized step, see [Example of a Parameterized Test](#) on page 212.

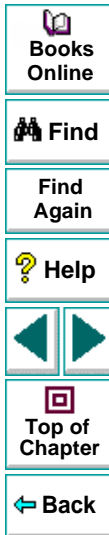
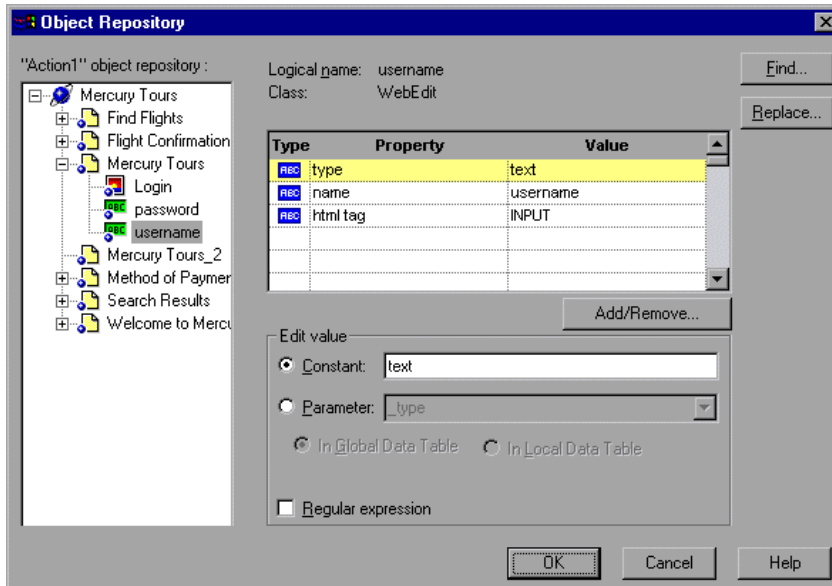
Parameterizing an Object in a Step

You can parameterize the object that you navigate in a step. For example, your Web site may include a form in which the user can choose to click one of several buttons. You may want to test how your site responds when different buttons are selected. Rather than record a separate test for clicking each button, you can parameterize your test so that during each iteration of the test run, Astra QuickTest clicks a different button.



To parameterize the object in a step:



- 1 Right-click a step in the test tree and choose **Object Repository**. The Object Repository dialog box opens and displays the properties of the object in the step.



The Object Repository displays information about the object in the step:

Information	Description
Repository Tree	Displays the object in its location within the object repository tree.
Logical name	The logical name of the object.
Class	The type of object. In this example, the "WebEdit" class indicates that the object is an edit box.

The dialog box displays the default properties you can parameterize, in a pane listing the properties, their values, and their types:


Pane Element	Description
Type	The  icon indicates that the property value is a constant. The  icon indicates that the property value is a parameter.
Property	The name of the property whose value will be parameterized.
Value	The value of the property to parameterize.
Add/Remove Properties	Opens the Add/Remove Properties dialog box, to enable you to modify the list of properties that you can parameterize. To add/remove a property, select/clear a check box and click OK . To set the default, click the Default button and click OK .

- Click the property to parameterize in the **Properties** section. The property is highlighted.



- 3 In the **Edit value** section, click **Parameter**.
- 4 In the **Parameter** 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.
- 5 Click **In Global Data Table** or **In Local Data Table**.
 - To add the parameter to the Global tab in the Data pane, click **In Global Data Table**.
 - To add the parameter to the Action tab, click **In Local Data Table**.

For more information on actions, see Chapter 14, [Working with Actions](#).
- 6 If you want to set the property value of the step as a regular expression, select the **Regular expression** check box. For more information, see Chapter 13, [Using Regular Expressions](#).
- 7 Click **Close** to save the parameter and close the dialog box.
- 8 If you created a new parameter, the **Astra Parameters** dialog box prompts you to add the new parameter to the table in the Data pane. Click **OK**. 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 the parameter by entering them directly into the table in the Data pane. For more information, see Chapter 15, [Working with Data Tables](#).

For more information about the Object Repository dialog box, see Chapter 3, [Creating Tests](#).



Books
Online



Find

Find
Again



Help



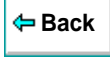
Top of
Chapter



Back

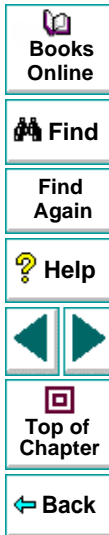
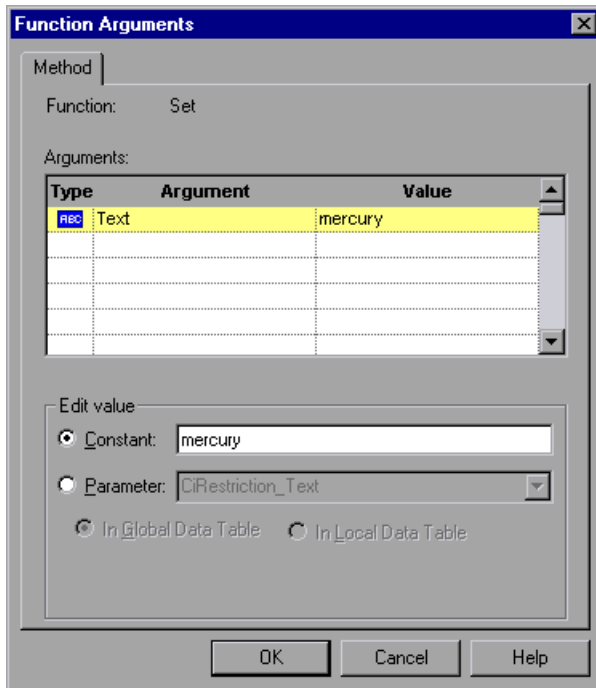
Parameterizing a Method in a Step

You can parameterize the method you use to navigate a step. For example, your Web site may 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 your test so that during each iteration of the test run, Astra QuickTest enters a different text string into the edit field.



To parameterize the method in a step:



- 1 Right-click a step in the test tree and choose **Function Arguments**. The Function Arguments dialog box opens and displays the method arguments in the step.



The Method tab displays the name of the function performed in the step:

Information	Description
Function	The name of the function performed.

The dialog box displays the default arguments you can parameterize, in a pane listing the arguments, their values, and their types:

Pane Element	Description
Type	The  icon indicates that the argument value is a constant. The  icon indicates that the argument value is a parameter.
Argument	The name of the argument whose value will be parameterized.
Value	The value of the argument to parameterize.

- 2 Click an argument in the **Arguments** section. The argument is highlighted.
- 3 In the **Edit value** section, click **Parameter**.
- 4 In the **Parameter** 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.



5 Click **Global** or **Local**.


- To add the parameter to the Global tab in the Data pane, click **Global**.
- To add the parameter to the Action tab, click **Local**.

For more information on actions, see Chapter 14, [Working with Actions](#).

6 If you want to set the argument value of the step as a regular expression, select the **Regular expression** check box. For more information, see Chapter 13, [Using Regular Expressions](#).

7 Click **Close** to save the parameter and close the dialog box.

8 If you created a new parameter, the Astra Parameters dialog box prompts you to add the new parameter to a table in the Data pane. Click **OK**. 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 the parameter by entering them directly into the table in the Data pane. For more information, see Chapter 15, [Working with Data Tables](#).

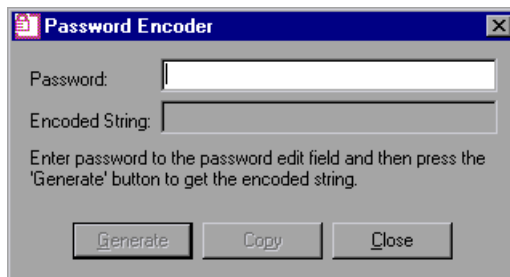


Encoding Passwords

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:

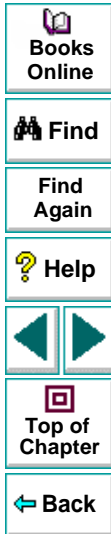
- 1 Select **Start > Programs > Astra QuickTest > Tools > Password Encoder**. The Password Encoder dialog box opens.



- 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.



Parameterizing Checkpoints

You can parameterize a checkpoint while recording your test or afterward. For information on parameterizing checkpoints while creating them, see Chapter 5, [Creating Checkpoints](#).

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 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, Astra QuickTest checks the flight information for a different destination. For an example of a parameterized checkpoint, see [Example of a Parameterized Test](#) on page 212.

To parameterize a checkpoint either while recording your test or afterward:

- 1 Right-click a checkpoint in the test tree and choose **Function Arguments**.
 - 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 Table Checkpoint Properties dialog box opens.
 - For a database checkpoint, the Edit Database Checkpoint dialog box opens.



Books
Online



Find

Find
Again



Help



Top of
Chapter




Back

- 2 In the dialog box, select the value to parameterize (if applicable) and click **Parameter** to set the value as a parameter.
- 3 In the **Parameter** 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.

You can create the parameter in the global or a local data table. For additional information, see Chapter 15, [Working with Data Tables](#).

- 4 To use a regular expression with the parameter, select the **Regular expression** check box. For additional information, see Chapter 13, [Using Regular Expressions](#).
- 5 Click **OK** to save the parameter and close the dialog box.
- 6 If you created a new parameter, the Astra Parameters dialog box prompts you to add the new parameter to the data. Click **OK**. A new column is highlighted in the table for the new parameter.

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



Note: You can specify additional data values for the parameter by entering them directly into the table in the Data pane. For more information, see Chapter 15, [Working with Data Tables](#).



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Example of a Parameterized Test

The following example shows how to parameterize a step object, step method, and checkpoint.

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, Astra QuickTest checks the flight information for a different locations.



Books
Online



Find

Find
Again



Help

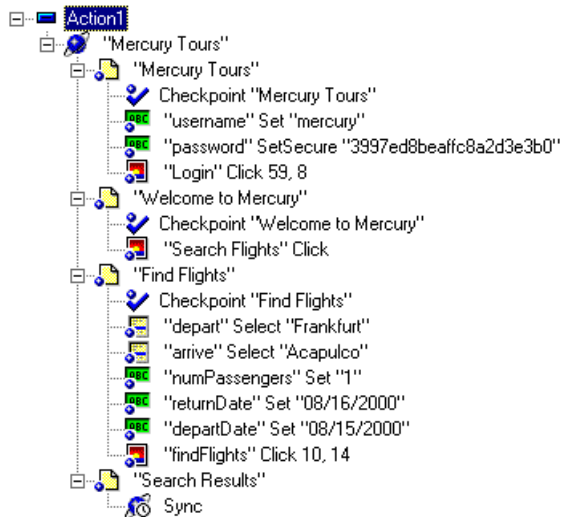


Top of
Chapter



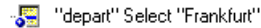
Back

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

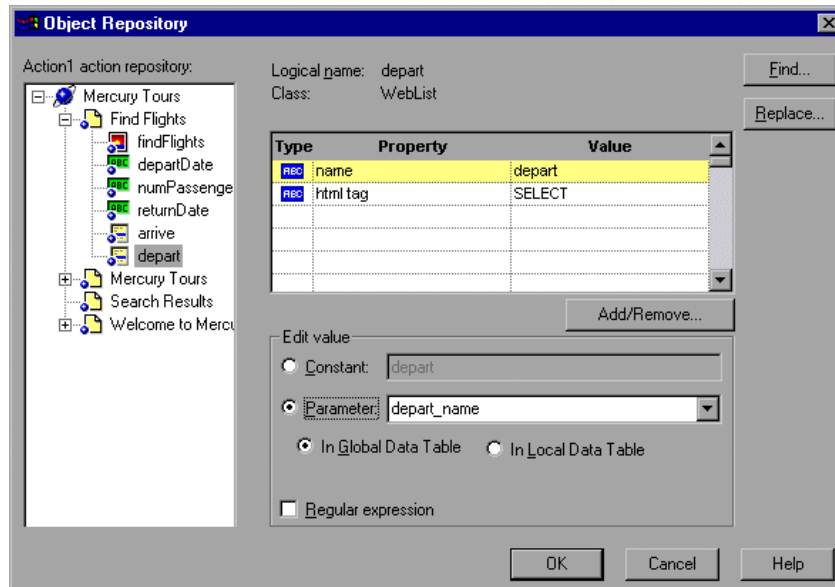


Parameterize a Step

Parameterize the object and the method of the following step:



In the Object Repository dialog box, select the “name” property. In the Parameter box, rename `depart_name` to `Activity`. Close the dialog box. The Activity column is added to the data table.



Books Online

Find

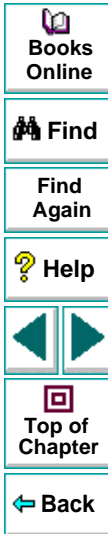
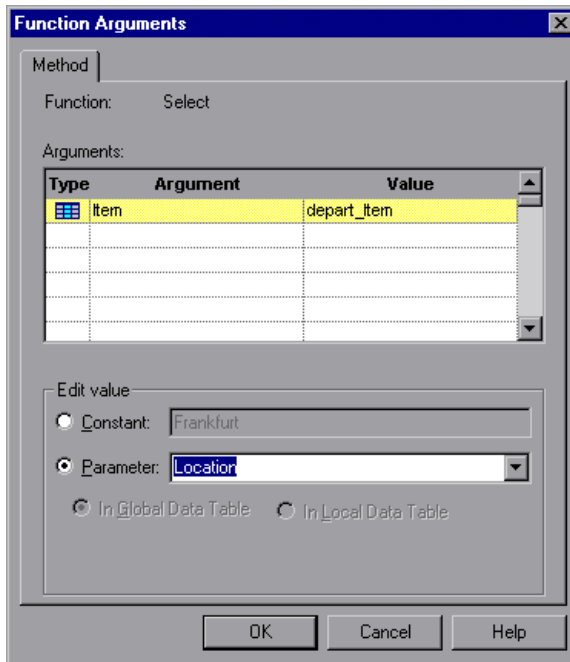
Find Again

Help

Top of Chapter

Back

In the following example, parameterize the method of the above step. In the Function Arguments dialog box, select the “item” property. In the Parameter box, rename the “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 Steps](#) on page 198.

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 following text:

search results
FLIGHTS

Flight departing from **Frankfurt to Acapulco** on
04/06/2000

Flight	Departure time	Cost
<input checked="" type="radio"/> Blue Sky Air 100	8am	\$ 716
<input type="radio"/> Blue Sky Air 101	1pm	\$ 637
<input type="radio"/> Blue Sky Air 102	5pm	\$ 677
<input type="radio"/> Blue Sky Air 103	11pm	\$ 585

continue... start over

Books
Online

Find

Find
Again

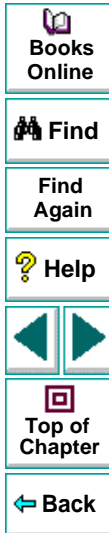
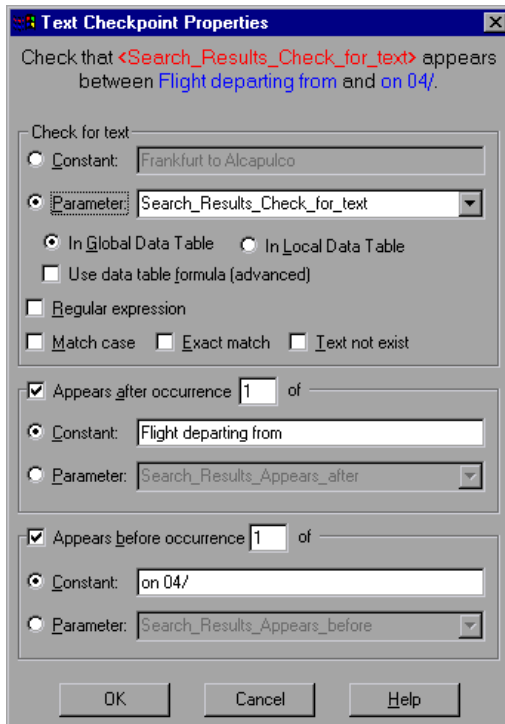
Help



Top of
Chapter

Back

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



For more information on parameterizing a checkpoint, see [Parameterizing Checkpoints](#) on page 209.

Enter Data in the Data Table

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

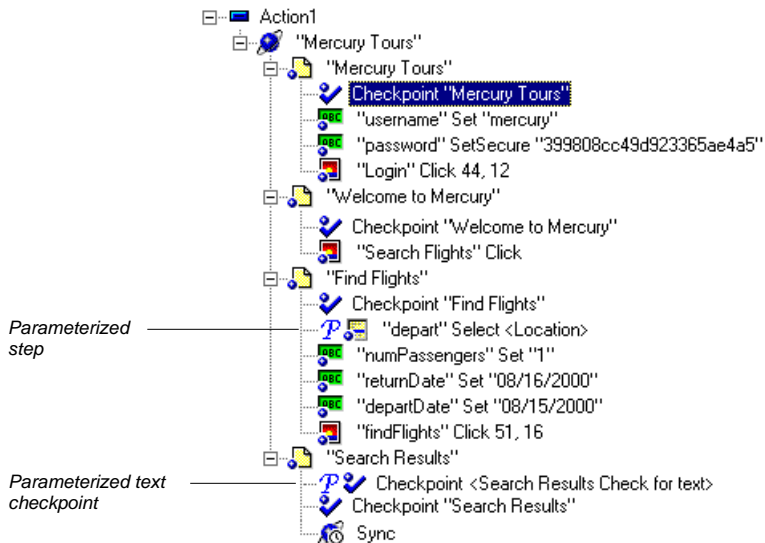
	Activity	Location	Search Results	Check for text
1	depart	Frankfurt	Frankfurt to Acapulco	
2	depart	Acapulco	Acapulco to Acapulco	
3	arrive	Acapulco	Acapulco to Acapulco	
4	arrive	Frankfurt	Acapulco to Frankfurt	
5				
6				
7				

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.



-  Books Online
-  Find
- Find Again
-  Help
- 
-  Top of Chapter
-  Back

Creating Tests

Creating Output Parameters

Astra QuickTest enables you to parameterize your test by retrieving a variable value from your test and entering it in your table in the Data pane, as an output parameter. You can subsequently use this output parameter 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 Parameters](#)
- [Creating Text Output Parameters](#)
- [Creating Object Output Parameters](#)
- [Creating Table Output Parameters](#)



About Creating Output Parameters

You parameterize your test by adding values to a table in the Data pane that replace variables in the test. When you run the test, Astra QuickTest runs one iteration of the test for each set of values from your table, as discussed in Chapter 11, [Parameterizing Tests](#).

You can also use output parameters to parameterize your test. An *output parameter* is a value that is retrieved while the test runs and is entered into your table.

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 parameter for the unique order number that the site generates when creating a new reservation. In the view reservation screen, you parameterize the order number input field. You use the same parameter with the unique order number that was previously retrieved as an output parameter.



When you run the test, Astra QuickTest retrieves the unique order number generated by the site for the new reservation and inserts it in the table for the order number output parameter. When the test reaches the order number input field required to view the reservation, Astra QuickTest uses the unique order number stored in the table for the order number input field parameter.



You can add an output parameter 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 parameters retrieved during a test run in the Runtime Data table. For more information, see [Viewing the Runtime Data Table for a Parameterized Test](#) on page 369.



Creating Page Output Parameters

When you create a page output parameter, you parameterize a constant page property by replacing it with a variable. When you run the test, Astra QuickTest retrieves the value in the output parameter and inserts it in the data table. For example, the number of links on a page may vary based on the selections a user makes on a form on the previous page. You could make an output parameter to return the number of links on the page during each run. To parameterize a page property, you open the Page Output Parameter Properties dialog box.

Adding a Page Output Parameter

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

To create a page output parameter while recording:



- 1 Choose **Insert > Output Parameter**.

The mouse pointer turns into a pointing hand.

- 2 Click the page to parameterize. The Object Selection - Output Param Properties dialog box opens.



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

- 4 Specify the settings for the output parameter. For more information, see [Understanding the Page Output Parameter Properties Dialog Box](#) on page 225.



- 5 Click **OK** to close the Page Output Parameter Properties dialog box.
- 6 A message box informs you that the output parameter will be added to the table in the Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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

To create a page output parameter after recording:

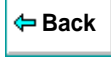


- 1 Make sure the **Display Views** button and the **ActiveScreen** tab are selected.
- 2 Click a page step in your test tree where you want to add an output parameter. The ActiveScreen displays the Web page corresponding to the highlighted step.
- 3 Right-click the page to parameterize in the ActiveScreen and choose **Output**. The Object Selection - Output Param Properties dialog box opens.



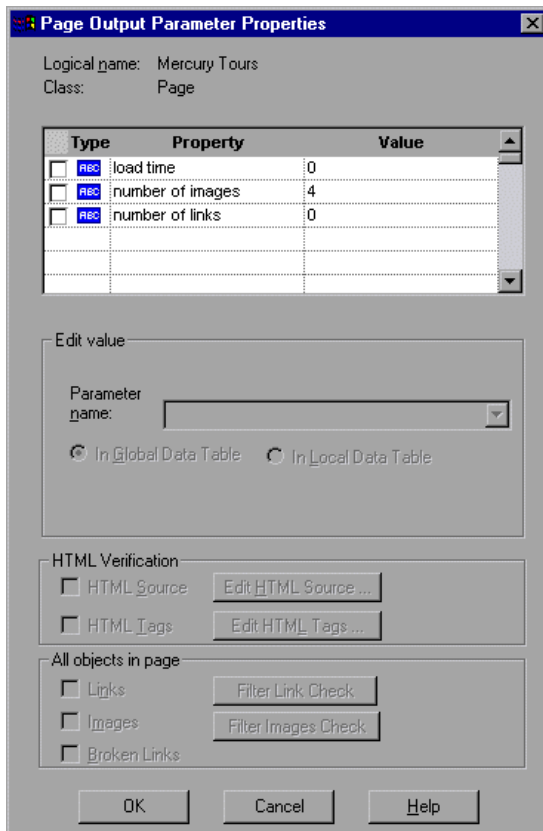
- 4 Select the page item and click **OK**. The Page Output Parameter Properties dialog box opens.
- 5 Specify the settings for the output parameter. For more information, see [Understanding the Page Output Parameter Properties Dialog Box](#) on page 225.
- 6 Click **OK** to close the Page Output Parameter Properties dialog box.
- 7 A message box informs you that the output parameter will be added to the table in the Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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



Understanding the Page Output Parameter Properties Dialog Box

In the Page Output Parameter Properties dialog box, you can specify which property of the page to parameterize.



Books Online

Find

Help

Top of Chapter



Identifying the Page

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

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

Choosing which Property to Parameterize

The dialog box also displays the page properties that you can parameterize, in a pane listing the properties, their values, and their types:

Pane Element	Description
Check box	To parameterize a property, select the corresponding check box.
Type	<p>The  icon indicates that the value of the property is a constant.</p> <p>The  icon indicates that the value of the property is a parameter.</p>



Pane Element	Description
Property	The name of the property to parameterize.
Value	The current value of the property. If you select to parameterize the property. The value column displays the parameter name.

Choosing an Output Parameter

In the Edit Value section, you use the following options to specify the output parameter name:

Option	Description
Parameter name	Specifies the output parameter name. Use the default output parameter name, or type a descriptive name for the output parameter.
In Global Data Table (default)	Adds the output parameter name to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .




Creating Text Output Parameters

When you create a text output parameter, you parameterize a constant text string by replacing it with a variable. To parameterize a text string, you open the Text Output Parameter Properties dialog box.

Adding a Text Output Parameter

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

To create a text output parameter while recording:

- 1 Highlight the text string you want to parameterize.
- 2 Choose **Insert > Text Output Parameter**.


The mouse pointer turns into a pointing hand.
- 3 Click the text string to parameterize. The Text Output Parameter Properties dialog box opens.
- 4 Specify the settings for the output parameter. For more information, see [Understanding the Text Output Parameter Properties Dialog Box](#) on page 230.
- 5 Click **OK** to close the Text Output Parameter Properties dialog box.
- 6 A message box informs you that the new output parameter will be added to the table in Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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





To create an output parameter after recording:

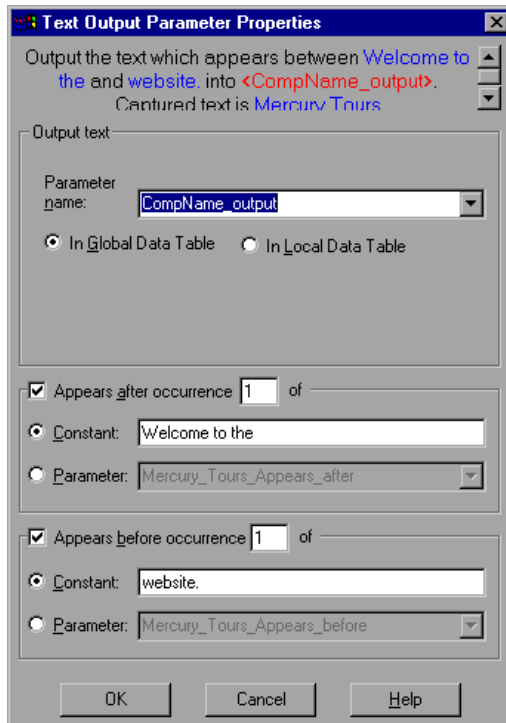
- 1 Make sure the **Display Views** button and the **ActiveScreen** tab are selected.
- 2 Click a step in your test where you want to create an output parameter.
The ActiveScreen displays the Web page corresponding to a highlighted step.
- 3 Highlight and then right-click the text string to parameterize in the ActiveScreen.
- 4 Choose **Insert Text Output**. The Text Output Parameter Properties dialog box opens.
- 5 Specify the settings for the output parameter. For more information, see [Understanding the Text Output Parameter Properties Dialog Box](#) on page 230.
- 6 Click **OK** to close the Text Output Parameter Properties dialog box.
- 7 A message box informs you that the new output parameter will be added to the table in the Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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



Understanding the Text Output Parameter Properties Dialog Box

In the Text Output Parameter Properties dialog box, you can specify which text to parameterize as well as which text appears before and after the parameter. This is particularly helpful when the text string you want to parameterize may appear several times in the same Web page.



Specifying which Text to Parameterize

In the Output Text section, you use the following options to specify the output parameter name for the highlighted text string:

Option	Description
Parameter name	Specifies the output parameter name. Use the default output parameter name, or type a descriptive name for the output parameter.
In Global Data Table (default)	Adds the output parameter name to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .



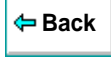
Specifying After which Text the Text to Parameterize Appears

In the Appears After section, you use the following options to specify which text, if any, should appear after the text output parameter:

Option	Description
<p>Appears after occurrence ___ of (default)</p>	<p>Outputs the text that appears after the text specified in the Constant or Parameter box.</p> <p>If the identical text string you specify appears more than once on the page, you can specify to which occurrence of the string you are referring.</p> <p>If you accept the default text that Astra QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate.</p> <p>If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to output the text that appears after the third occurrence of the words "Mercury Tours", enter 3 in the Appears after occurrence box.</p> <p>To output text starting from the beginning of the page, clear this check box.</p>



Option	Description
Constant (default)	Displays the text after which the text to output appears. Tip: If you modify the text, use a unique string whenever possible so that the occurrence number is always 1.
Parameter	Sets the text string as a parameter.



Specifying Before which Text the Text to Parameterize Appears

In the Appears Before section, you use the following options to specify which text, if any, should appear before the text output parameter:

Option	Description
<p>Appears before occurrence ___ of (default)</p>	<p>Outputs the text that appears before the specified text in the Constant or Parameter box.</p> <p>Astra QuickTest starts counting occurrences of the Appears before text you specify, from the end of the Appears after string. In other words, it starts looking for the specified text from the text you selected to check.</p> <p>If you accept the default text that Astra QuickTest recommends, the number in the dialog box will be correct. If you modify the recommended text string and the string you specify appears in the text to output as well as in the Appears before text, you need to modify the occurrence number accordingly.</p> <p>To output all text on the page that appears after the text specified in the Appears After option, clear this check box.</p>



Option	Description
Constant (default)	Displays the text before which the text to check should appear. Tip: If you modify the text, use a unique string whenever possible so that the occurrence number is always 1.
Parameter	Sets the text string as a parameter.



Creating Object Output Parameters

You can parameterize an object on your Web page to create an object output parameter. To parameterize an object, you open the Object Output Parameter dialog box.

Adding an Object Output Parameter

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

To create an object output parameter while recording:



- 1 Choose **Insert > Output Parameter**.

The mouse pointer turns into a pointing hand.

- 2 Click the object to parameterize. The Object Selection - Output Param Properties dialog box opens.
- 3 Select the tree item you want to parameterize. The tree item name depends on the object's class, for example:

Object	Class
Check box	WebCheckBox
Edit box	WebEdit



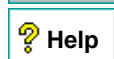
Object	Class
Image	Image
Radio button	WebRadio

- 4 Click **OK**. The Output Param Properties dialog box opens.
- 5 Specify the settings for the output parameter. For more information, see [Understanding the Output Param Properties Dialog Box](#) on page 239.
- 6 Click **OK** to close the Output Param Properties dialog box.
- 7 A message box informs you that the output parameter will be added to the table in the Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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

To create an object output parameter after recording:

- 1 Make sure the **Display Views** button and the **ActiveScreen** tab are selected.
- 2 Click a step in your test where you want to create an output parameter.
The ActiveScreen displays the Web page corresponding to a highlighted step.
- 3 Right-click the object to parameterize in the ActiveScreen and choose **Output**. The Object Selection - Output Param Properties dialog box opens.



- 4 Select the tree item you want to parameterize. The tree item name depends on the object's class, for example:

Object	Class
Check box	WebCheckBox
Edit box	WebEdit
Image	Image
Radio button	WebRadio

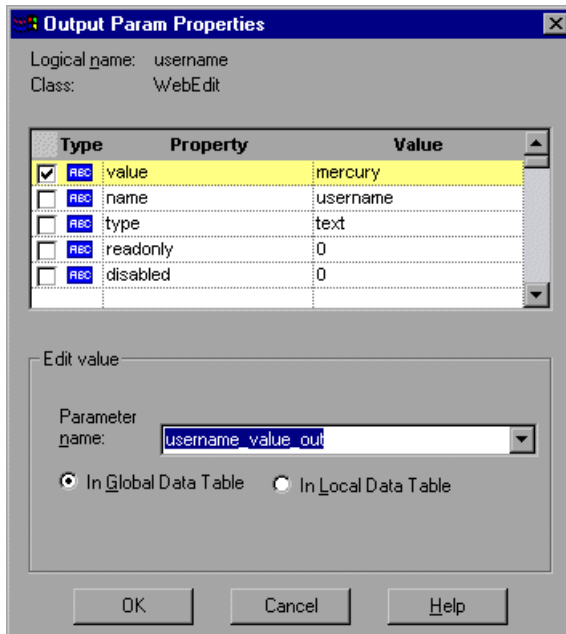
- 5 Click **OK**. The Output Param Properties dialog box opens.
- 6 Specify the settings for the output parameter. For more information, see [Understanding the Output Param Properties Dialog Box](#) on page 239.
- 7 Click **OK** to close the Output Parameter Properties dialog box.
- 8 A message box informs you that the output parameter will be added to the table in the Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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



Understanding the Output Param Properties Dialog Box

In the Output Parameter Properties dialog box, you can specify which property of the object to parameterize.





Identifying the Object

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

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 "WebEdit" class indicates that the object is an edit field.

Choosing which Property to Parameterize

The dialog box also displays the properties of the object you can parameterize, in a pane listing the properties, their values, and their types:

Pane Element	Description
Check box	To parameterize a property, select the corresponding check box.
Type	<p>The  icon indicates that the value of the property is a constant.</p> <p>The  icon indicates that the value of the property is a parameter.</p>



Pane Element	Description
Property	The name of the property to check.
Value	The value of the property. If you choose to parameterize the property, the Value box displays the parameter name.

Choosing an Output Parameter

In the Edit Value section, you use the following options to specify the output parameter name:

Option	Description
Parameter name	Specifies the output parameter name. Use the default output parameter name, or type a descriptive name for the output parameter.
In Global Data Table (default)	Adds the output parameter name to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .



Creating Table Output Parameters

You can parameterize a text string in your table to create a table output parameter. To parameterize a text string in a table, you open the Table Output Parameter Properties dialog box.

Adding a Table Output Parameter

You can create a table output parameter while recording your test or afterward.

To create a table output parameter while recording:



- 1 Choose **Insert > Output Parameter**.

The mouse pointer turns into a pointing hand.

- 2 Click the table to parameterize. The Object Selection - Output Param Properties dialog box opens.



- 3 Select a table item and click **OK**. The Table Output Parameter Properties dialog box opens.
- 4 Specify the settings for the output parameter. For more information, see [Understanding the Table Output Parameter Properties Dialog Box](#) on page 244.
- 5 Click **OK** to close the Table Output Parameter Properties dialog box.



- 6 A message box informs you that the output parameter will be added to the table in the Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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

To create a table output parameter after recording:



- 1 Make sure the **Display Views** button and the **ActiveScreen** tab are selected.
- 2 Click a step in your test where you want to create an output parameter.

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

- 3 Highlight a text string in a table on the ActiveScreen tab.
- 4 Right-click the table to parameterize in the ActiveScreen and choose **Output**. The Object Selection - Output Parameter Properties dialog box opens.



- 5 Select a table item and click **OK**. The Table Output Parameter Properties dialog box opens.
- 6 Specify the settings for the output parameter. For more information, see [Understanding the Table Output Parameter Properties Dialog Box](#) on page 244.

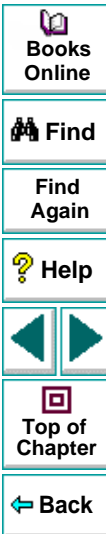
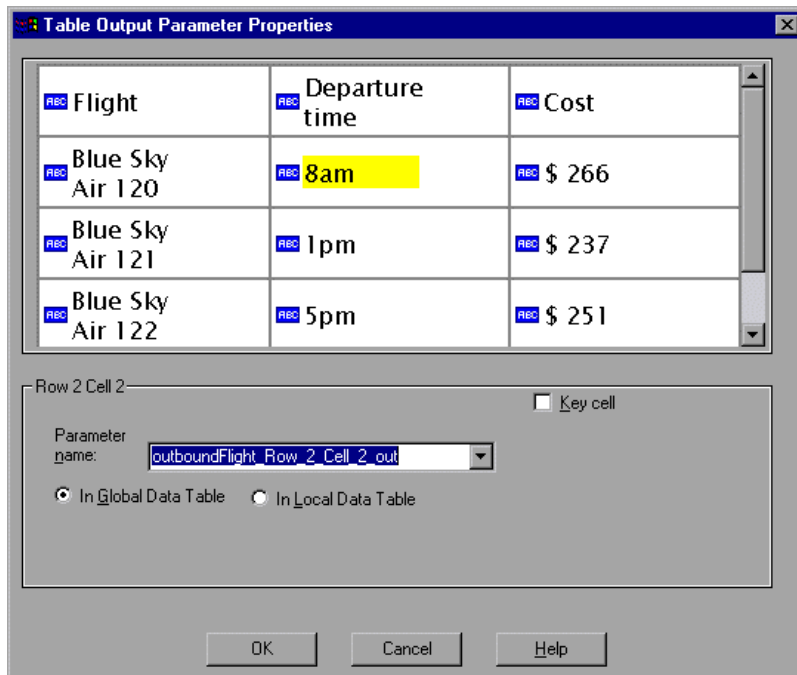
- 7 Click **OK** to close the Table Output Parameter Properties dialog box.
- 8 A message box informs you that the output parameter will be added to the table in the Data pane. Click **OK**. A new column is highlighted in the table for the new output parameter.

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



Understanding the Table Output Parameter Properties Dialog Box

In the Table Output Parameter Properties dialog box, you can specify the name of the output parameter and/or you can assign a cell as a key cell.



The dialog box displays rows and columns of a table. Your highlighted text string appears in a cell.

Use the following options to specify the output parameter name:

Option	Description
Parameter name	Specifies the output parameter name. To use an output parameter that you already created, select an output parameter from the list. To create a new output parameter, you can use the default output parameter name, or type a descriptive name for the output parameter.
In Global Data Table (default)	Adds the output parameter name to the Global tab in the Data pane. For more information, see Chapter 14, Working with Actions .
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see Chapter 14, Working with Actions .

You can also mark cells as key cells if you want them to be searched by content rather than by row number. You cannot, however, assign the same cell as an output parameter and a key row.

Tip: If you want to mark one cell in a table as a key cell and another as an output parameter, then mark the key cell before you mark the output parameter cell.



Creating Tests 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 Astra 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 (\), Astra QuickTest searches for the literal character.



Using Regular Expressions in Steps

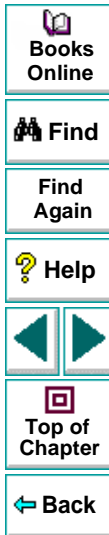
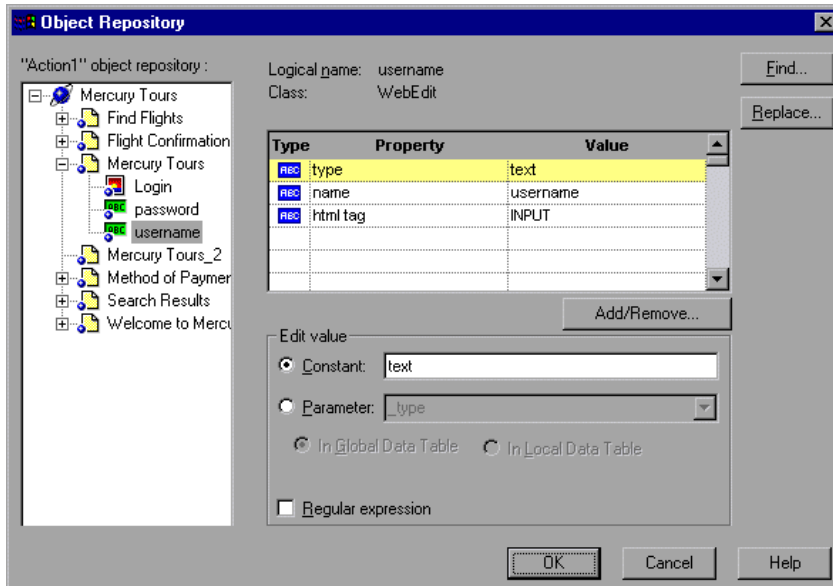
You can use regular expressions when defining or parameterizing a step in your test tree. A step is made up of an *object* that you navigate in your Web page, and/or a *method* by which you navigate the step. You can use regular expressions when defining or parameterizing the object of a step.

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 reappears 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 Astra 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 Repository**. The Object Repository dialog box opens.



The dialog box displays information about the object in the step:

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 "WebButton" class indicates that the object is a button.

The dialog box displays the properties of the object in the step:

Option	Description
Type	The  icon indicates that the property value is a constant. The  icon indicates that the property value is a parameter.
Property	The name of the property value.
Value	The value of the property.
Add/Remove	Opens the Add/Remove dialog box to enable you to modify the list of properties that you can check. To add/remove a property, select/clear a check box and click OK . To set the default, click the Default button and click OK .

- 2 Click the property you want to set as a regular expression in the **Properties** section. The property is highlighted.



3 In the **Edit value** section, set the property value as a regular expression.

- To set the property value as a constant, click **Constant**.

In the **Constant** box, Enter the regular expression syntax for the string. For information on regular expression syntax, see [Regular Expression Syntax](#) on page 261.

- To set the property value as a parameter, click **Parameter**.

In the **Parameter** 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 11, [Parameterizing Tests](#).

To add the parameter to the Global tab in the Data pane, select **In Global Data Table**. To add the parameter to the Action tab, select **In Local Data Table**. For more information, see Chapter 14, [Working with Actions](#).

Note: The property value in the **Parameter** 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 261. For information on editing the table, see Chapter 11, [Parameterizing Tests](#).




- 4 Select the **Regular Expression** check box.
- 5 You are prompted to add a backslash (/) before each special character in the **Constant** text box to treat it literally.

By default, Astra QuickTest treats all characters in a regular expression literally, except for a period (.), 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 (\), Astra QuickTest treats it as literal character.

- Click **Yes** to instruct Astra QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
 - Click **No** to instruct Astra QuickTest to treat all the characters literally, except for special characters.
- 6 Click **OK** to save and close the Repository Editor dialog box.

If you created a new parameter, the Astra parameters dialog box prompts you to add the new parameter to the table in the Data pane. Click **OK**. 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.



Using Regular Expressions in Object Checkpoints

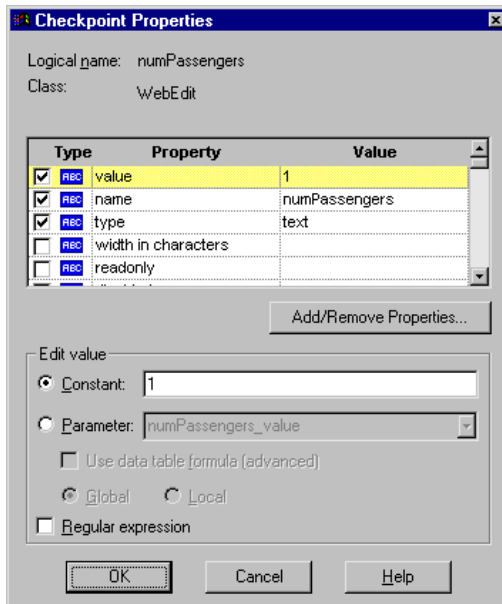
When creating an object checkpoint to verify that an object appears on your Web site, you can also set the property value of the object as a regular expression, so that an object with a varying name can be verified.

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. Astra 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 on 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 5, [Creating Checkpoints](#).



- 3 Select the check box of a property to be set as a regular expression. The property is highlighted.
- 4 In the **Edit value** section, set the property value as a regular expression.
 - To set the property value as a constant, click **Constant**.

In the **Constant** box, set the value as a regular expression. For information on regular expression syntax, see [Regular Expression Syntax](#) on page 261.

- To set the property value as a parameter, click **Parameter**.

In the **Parameter** 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 11, [Parameterizing Tests](#).

To add the parameter to the Global tab in the Data pane, select **Global**. To add the parameter to the Action tab, select **Local**. For more information, see Chapter 14, [Working with Actions](#).

Note: The property value in the **Parameter** 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 261. For information on editing the table, see Chapter 11, [Parameterizing Tests](#).





- 5 Select the **Regular Expression** check box.
- 6 You are prompted to add a backslash (\) before each special character in the **Constant** text box to treat it literally.

By default, Astra 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 (\), Astra QuickTest treats it as literal character.

- Click **Yes** to instruct Astra QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
 - Click **No** to instruct Astra QuickTest to treat all the characters literally, except for special characters.
- 7 Click **OK** to save and close the Checkpoint Properties dialog box.

If you created a new parameter, the Astra parameters dialog box prompts you to add the new parameter to the table in the Data pane. Click **OK**. 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. The  icon indicates a checkpoint.



Using Regular Expressions in Text Checkpoints

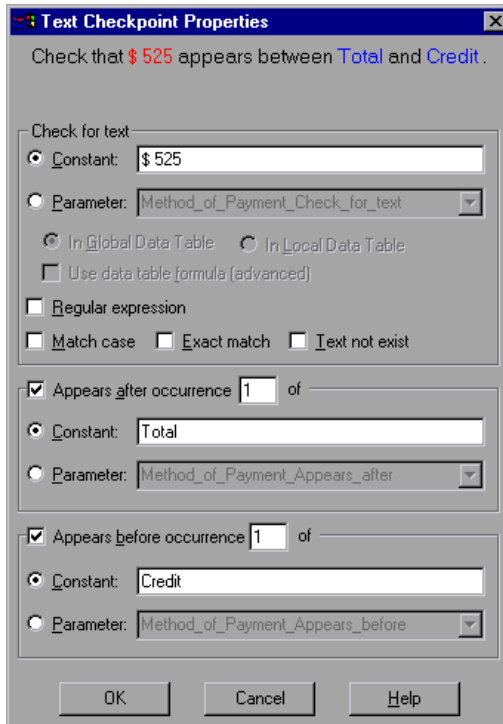
When creating a text checkpoint to check that a varying text string appears on your Web site, you define the text string as a regular expression.

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 Astra QuickTest will ignore variations in the text string as long as the value is not less than \$300.

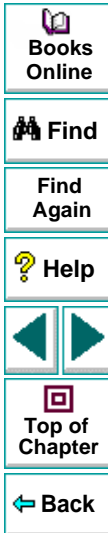


To define a regular expression in a text checkpoint:

- 1 Open the Text Checkpoint Properties dialog box.



The Text Checkpoint Properties dialog box enables you to specify which text to check as well as which text appears before and after the text to check. For more information, see Chapter 5, [Creating Checkpoints](#).



2 In the **Check for text** section, define the text string as a regular expression.

- To set the text string as a constant, click **Constant**.

In the **Constant** box, define the text string as a regular expression. For information on regular expression syntax, see [Regular Expression Syntax](#) on page 261.

- To set the text string as a parameter, click **Parameter**.

In the **Parameter** 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 11, [Parameterizing Tests](#).

To add the parameter to the Global tab in the Data pane, select **Global**. To add the parameter to the Action tab, select **Local**. For more information, see Chapter 14, [Working with Actions](#).

Note: The name in the **Parameter** 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 261. For information on editing the table, see Chapter 11, [Parameterizing Tests](#).





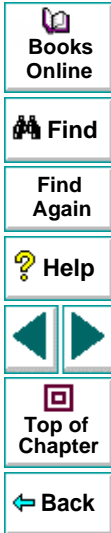
- 3 Select the **Regular Expression** check box.
- 4 You are prompted to add a backslash (\) before each special character in the **Constant** text box to treat it literally.

By default, Astra 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 (\), Astra QuickTest treats it as literal character.

- Click **Yes** to instruct Astra QuickTest to treat a special character literally. The special character is now preceded by a backslash (\).
 - Click **No** to instruct Astra QuickTest to treat all the characters literally, except for special characters.
- 5 Enter the regular expression syntax for the string in the **Constant** text box, as described in [Regular Expression Syntax](#) on page 261.
 - 6 Click **OK** to save and close the Text Checkpoint Properties dialog box.

If you created a new parameter, the Astra parameters dialog box prompts you to add the new parameter to the table in the Data pane. Click **OK**. 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. The  icon indicates a checkpoint.



Regular Expression Syntax

Astra QuickTest searches for 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 (\) as described below. When one of these special characters is preceded by a backslash (\), Astra QuickTest searches for the literal character.

The following options can be used to create regular expressions:

Matching Any Single Character

A period (.) instructs Astra 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 appears as the first or last character within brackets, or after a caret (^).

A caret (^) instructs Astra 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 appears 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.



Books
Online



Find



Find
Again



Help



Top of
Chapter



Back

Matching Specific Characters

You can use special characters to match zero, one, or more occurrences of the preceding character.

Matching Zero or More of the Preceding Character

An asterisk (*) instructs Astra QuickTest to match zero or more occurrences of the preceding character. For example:

`ca*r`

matches `car`, `caaaaaar`, and `cr`.

Matching One or More of the Preceding Character

A plus sign (+) instructs Astra 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 of the Preceding Character

A question mark (?) instructs Astra QuickTest to match zero or one occurrences of the preceding character. For example:

`ca?r`

matches `car` and `cr`, but nothing else.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Matching Either the Preceding or Following Expression

A vertical line (|) instructs Astra QuickTest to match either the preceding or following expression. For example:

`l|book`

matches `l` or `book`, while the following expression:

`(l|b)ook`

matches `look` or `book`.

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 Astra 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`.

Matching character class operators

Brackets surrounding opening and closing colons ([: :]) instructs Astra 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 Astra 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`

Grouping Regular Expressions

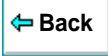
Parentheses (`()`) instruct Astra QuickTest to match the pattern and remember the match. The matched substring can be retrieved using `[0]...[n]`.

Using the Backslash Character

A backslash (`\`) instructs Astra QuickTest to treat the next character as either a special character or a literal. Examples are as follow:



- `n` matches the character `n`
- `\n` matches a newline character
- `\\` matches `\`
- `\(` matches `(`



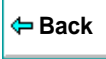
Creating Tests

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 List**
- **Setting Action Properties**
- **Creating New Actions**
- **Inserting Existing Actions**
- **Calling an Action from within 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 may also *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 actions** - actions that can only be used in the test in which it was created, and only once.
- **reusable actions** - actions that can be called multiple times by the test in which it was created as well as by other tests.
- **external actions** - reusable actions 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* (an action created in another test) can be inserted as a non-reusable copy of the original action, or 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 287. For more information about inserting existing actions see [Inserting Existing Actions](#) on page 290.

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.



When you divide your test into multiple actions, you can parameterize each one separately. For every action in your test, Astra QuickTest creates a corresponding sheet in the Data pane so that you can enter parameters that are specific to that action only. For more information on parameterizing actions, see [Parameterizing Actions](#) on page 272. For information on parameterizing tests, see Chapter 11, [Parameterizing Tests](#), and Chapter 12, [Creating Output Parameters](#).

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](#).



Parameterizing Actions

You parameterize an action by parameterizing the steps within that action. When you add a parameter to your test, you specify whether it is a *global* parameter, for the entire test, or a *local* parameter, for a specific action within the test.

In the parameterization dialog boxes:

- Choosing **In Global Data Table** 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 **In Local Data Table** creates parameters in the corresponding action sheet in the Data pane. When there are parameters in an Action sheet, you can set Astra 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 284.

Note: If you create local 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 local data sheet, or to run iterations only on the rows you specify. For more information about the Run tab settings, see [page 284](#).



For more information on parameterizing, see Chapter 11, [Parameterizing Tests](#).

For more information on the data table, see Chapter 15, [Working with Data Tables](#).

When you create a 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 11, [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:

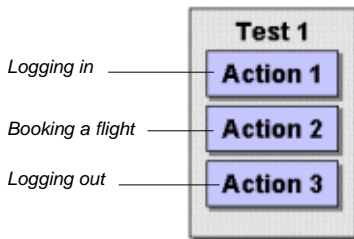
- 1 The travel agent logs into the flight reservation system.
- 2 The travel agent books five sets of customer flight itineraries.
- 3 The travel agent logs out of the flight reservation site.



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:

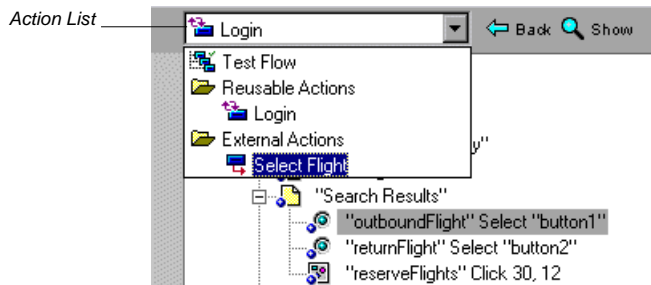


Using the Action List

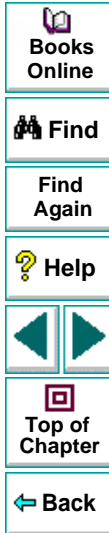
By default, Astra QuickTest opens with the Action List hidden in the Tree View. You can choose to view it at any time by enabling the **Show Action List in Tree View** option from the General tab of the Options dialog box.

For more information, see Chapter 24, [Setting Astra QuickTest Testing Options](#).

The first time you insert a reusable or external action in a test, the **Show Action List in Tree View** option is automatically enabled and the Action List is displayed above the test tree.



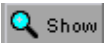
The Action List enables you to view either the entire *test flow* or a selected action view. The *test flow* displays the overall flow of your test with all the actions in your test. An *action view* displays all the details of the selected action.



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 [Making Actions Reusable](#) on page 282.

There are three ways to switch to the action view for a reusable or external action:

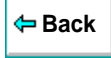
- Double-click the action you want to view.
- Highlight the action you want to view and click the Show button.
- Select the name of the action from the Action List.



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 294.

If you are working in a test without reusable or external actions, you can hide the Action List. To hide the Action List, clear the **Show Action List in the Tree View** option in the General Tab of the Options dialog box.

For more information, see Chapter 24, [Setting Astra QuickTest Testing Options](#).



In the Expert view, the Action List 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](#).



Setting Action Properties

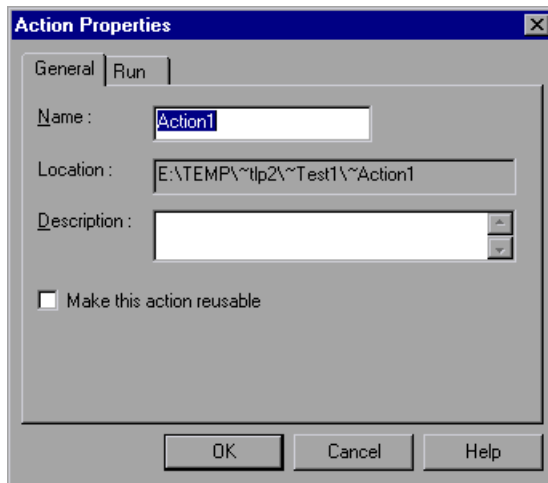
The Action Properties dialog box enables you to modify an action name, add a description for an action, set an action as a reusable action, and set the run properties for an action.

Opening the Action Properties Dialog Box

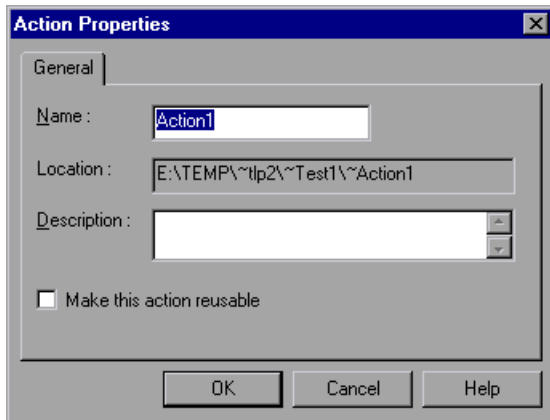
You can open the Action Properties dialog box from the Tree View with the test flow displayed, from the Tree View with an action view displayed or from the Expert View (displays the script of the current action).



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:



Adding or Editing an Action Description

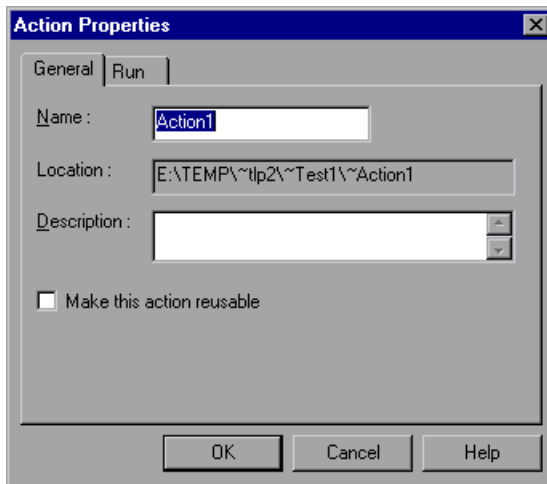
When you add a description to an action, the description is displayed in the action's Action Properties dialog box. An action description helps you and other testers know what a specific action does without reviewing the entire script or expanded tree of the action. The description also appears in the description section of the Insert Copy of Action and Insert Call to Action dialog boxes.



This enables you and other testers 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 290.

To add or edit an action description:

- 1 Right-click the action in the test tree or right-click anywhere in the Expert View and select **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 287.

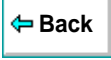
Making Actions Reusable

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.

Tip: Calling a reusable action as a link 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.

To make an action reusable:

- 1 Right click the action you want to change and select **Action Properties**. The Action Properties dialog box opens.
- 2 Select **Make this action a reusable action**.





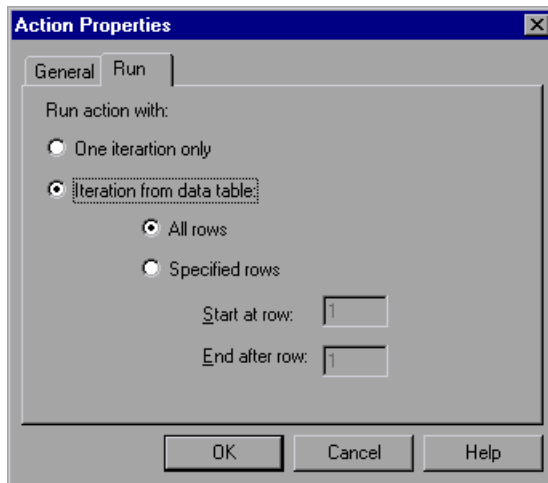
- 3 Click **OK**. If the action tree was expanded, it collapses, and the action icon changes to a reusable action icon.

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 List](#) on page 275.



Setting the Run Properties for an Action

You can use the Action Properties Run tab to instruct Astra 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 only certain rows in the data table.



The Run tab includes the following options:

Option	Description
<p>One iteration only</p>	<p>Runs the action only once, using the row that corresponds to the global iteration counter.</p> <p>For example, suppose an action's local 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's local parameters use during each repetition of the test are based on the iteration number for the test.</p> <p>During the first iteration of the test, local parameters in the action will take data from the first row of the local data sheet. In the second iteration of the test, local parameters in the action will take data from the second row of the local data sheet. In the third and subsequent iterations of the test, the local parameters in the action will continue to take data from the second (last) row of the data sheet.</p>
<p>Iterations from the data table</p>	<p>Runs the action with iterations on the rows specified in the next option.</p>

 Books Online
 Find
 Find Again
 Help
 
 Top of Chapter
 Back

Option	Description
All Rows	Runs the action with iterations using all rows in the action's local data table.
Specified Rows	Runs the action with iterations using the values in the action's local data table for the specified row range. Use Start at row to indicate the row number to start the run. Use End after row to indicate the row number to end the run.

Note: 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.

Note: The Run tab of the Action Properties dialog box applies to individual actions and refers to the rows in the action's local 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](#).



Creating New Actions

You can add new actions to your test during a recording session or before you begin a recording session.

To create a new action in your test:

- 1 If you are in the middle of a recording session, click **Stop** when you reach the point in your recording session where you want to start a new action.
- 2 If you want to insert the action within an existing action, click the step after which you want to insert the new action.
- 3 Choose **Insert > New Action** or click the **New Action** button. The Insert New Action dialog box opens.




4 Type a new action name or accept the default name.

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 [Calling an Action from within an Action](#) on page 297.

6 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 280.

7 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.



8 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 [Making Actions Reusable](#) on page 282.

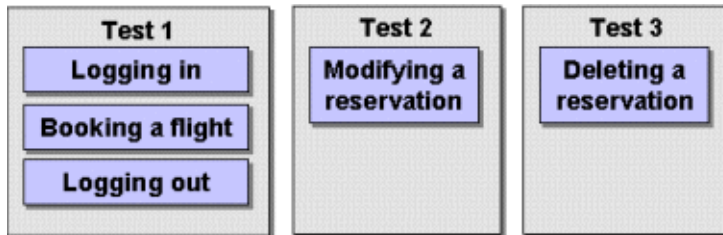


Inserting Existing Actions

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 it 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.

Inserting a Copy of an Action

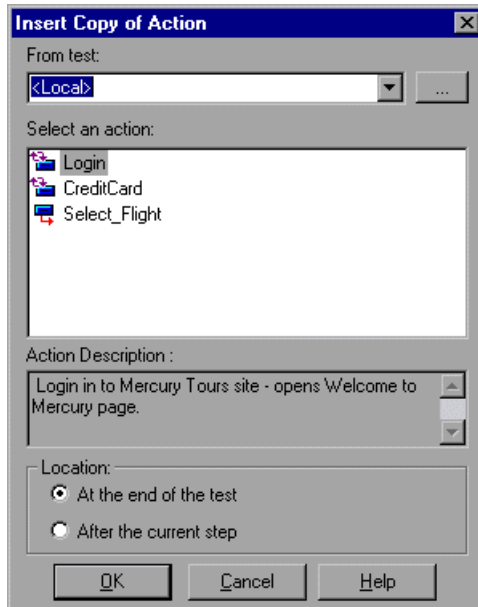
When you insert a copy of an action into a test, the action is copied in its entirety, including checkpoints, parameterization, and the corresponding action tab in the Data pane. The action is inserted into the test as an independent, non-reusable action (even if the original action was reusable). Once 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, appears in the Action Description box. This helps you identify the action you want to insert. For more information about action descriptions see [page 281](#).
- 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 [page 297](#).
- 5 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.

Note: If you try to insert an action in a test that already contains an action with the same name, you are prompted to rename the action you are inserting.



Inserting a Call to an Action

You can insert a call (link) to a reusable action from your current (local) test, or from an external test.

When you insert a call to an external action, the action is inserted in read-only format. This includes the data from the action's local data sheet in the data table. You can view the components of the action in the action view, but you cannot modify them.

To modify the action, you must open the test in which the action originated. If the original action, or the corresponding data from the data table is modified, the modifications apply to all tests that call that action.

Tip: You can view the location of the original action in the General tab of the Action Properties dialog box.



Books
Online



Find

Find
Again



Help



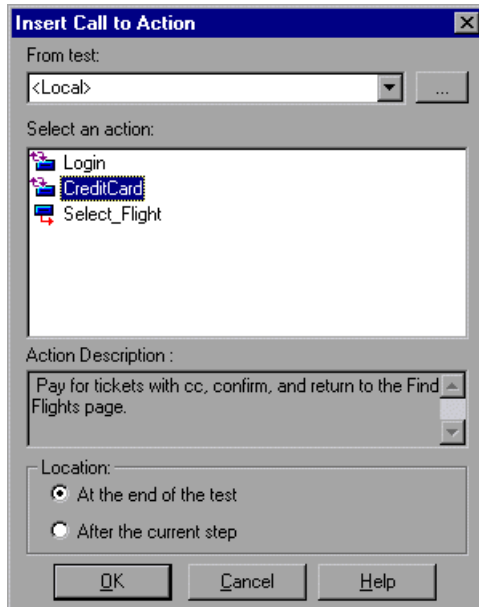
Top of
Chapter



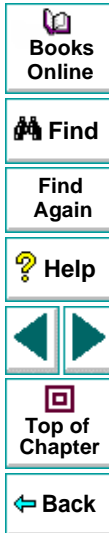
Back

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, appears in the Action Description box. This helps you identify the action you want to insert. For more information about action descriptions see [page 281](#).

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 [page 297](#).



5 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.

Note: If you try to insert an action in a test that already contains an action with the same name, you are prompted to rename the action you are inserting.



Books
Online



Find

Find
Again



Help



Top of
Chapter

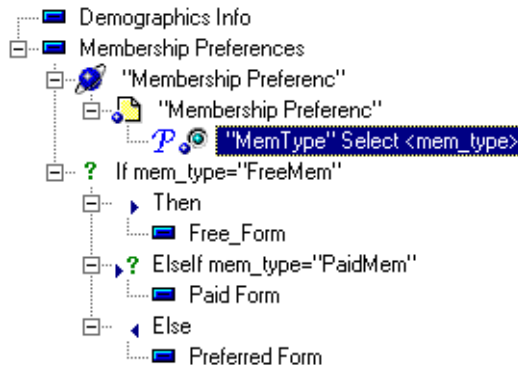


Back

Calling an Action from within an Action

Sometimes you may want to run an action within an action. This can help you maintain the modularity of your test. It also enables 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:



Books Online

Find

Find Again

Help

Top of Chapter

Back

For more information about inserting conditional statements, see [Using Conditional Statements](#) on page 412.

To insert an action within an 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 [page 287](#), or follow the instructions for inserting an existing action as described on [page 290](#).
- 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 appears within the parent action with the 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](#).



Removing Actions From a Test

The procedures and effects of removing non-reusable actions, calls to external actions, or local, reusable action are different.

- When you remove a non-reusable action, you delete the action entirely.
- When you remove a call to an external action, you remove the action from your test flow, but you do not affect the original action.
- When you remove a local, reusable action, you can either remove the selected call to the action from the test without affecting the action itself, or you can delete the action entirely.

Removing a Non-Reusable Action or a Call to an External Action

The procedure for removing non-reusable actions and calls to external actions is the same, even though the effect of the removal is different, as described above.

To remove a non-reusable action or call to an external action:

- 1 Select the action you want to remove and press the **Delete** key on your keyboard or select **Edit > Delete Action**. A delete confirmation message box opens.
- 2 Click **Yes** to confirm.



Removing a Call to the Reusable Action From the Test Flow

You should choose to remove a call to the reusable 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 Local data table tab for the action remains. You can still view and edit the action by selecting the action from the Action List in the Tree View or in the Expert View.

After you remove a call to an action, you can still 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 290.

To remove a call to a reusable 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 Action**, 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 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 Choose **Edit > Delete Action**, or press **Delete** on your keyboard. The Confirm Action Removal message box opens.
- 3 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 local (action-specific) 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, and books three flights and logs out, etc. To parameterize the ‘book a flight’ action, you choose **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 local 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 more detailed action descriptions. This facilitates inserting actions from one test to another. You can rename the action by choosing **Edit > Rename Action**.



Books
Online



Find

Find
Again



Help

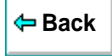


Top of
Chapter



Back

- 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](#).



Creating Tests

Working with Data Tables

Astra QuickTest enables you to create and run tests that are driven by data stored in the data table.

This chapter describes:

- **Global and Local Sheets**
- **Editing the Data Table**
- **Importing Data from a Database**
- **Using Formulas in the Data Table**
- **Using Data Table Scripting Functions**



About Working with Data Tables

You can parameterize your test with input and output parameters so that it will run several times on different sets of data. The data your test uses is stored in the data table, which appears 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 for a Parameterized Test](#) on page 369.



Global and Local Sheets

There are two types of sheets within the data table: *Global* and *Local*. 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 a Local 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 Local tab corresponding to that action.



Global Sheet

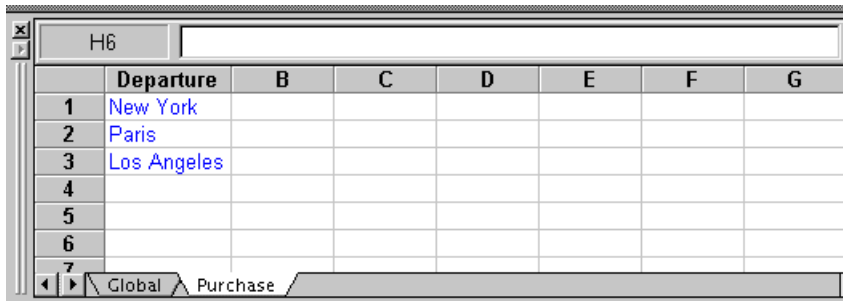
The Global sheet contains the data that replaces parameters in each iteration of the test. If you create a Global parameter called Arrivals, the Global sheet might look like this:

	Arrivals	B	C	D	E	F	G
1	San Francisco						
2	New York						
3	Paris						
4							
5							
6							
7							



Local Sheets

Each time you add a new action to the test, a new *Local* sheet is added to the data table. Local sheets are automatically labeled with the exact name of the corresponding action. The data contained in a local sheet is relevant for the corresponding action only. For example, if a test had the data table below, Astra QuickTest would only use the data contained in the *Departure* column when running iterations on the *Purchase* action:



	Departure	B	C	D	E	F	G
1	New York						
2	Paris						
3	Los Angeles						
4							
5							
6							
7							



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Editing the Data Table

The data table contains the values that Astra QuickTest substitutes for parameters when you run a test. Whenever you save your test, Astra QuickTest automatically saves the test's data table. Astra QuickTest automatically saves the data table for a test in the test folder and assigns it an .xls extension.

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.



	departure	arrival	C	D	E	F	G
1	Acapulco	New York					
2	New York	Paris					
3	London	Frankfurt					
4							
5							
6							
7							

- Each *row* in the table represents the set of values that Astra 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	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.
Export	Exports the table to a specified excel file.
Print	Prints the table.



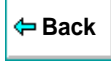
Edit Menu

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.
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.



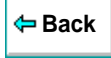
Edit Command	Description
Fill Down	Copies data in the top cell of the selected range to all cells below it within the selected range.
Find	Finds a cell containing specified text. You can search by row or column in the table and specify to match case or find entire cells only.
Replace	Finds a cell containing 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, across or down and Astra QuickTest automatically fills in the cells in the range according to the autofill list.
Import from	Enables you to import data from the specified source.



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: (M/d/yy)	Sets format to Date with the M/d/yy format.
Time: h:mm AM/PM	Sets format to Time with the h:mm AM/PM format.
Custom Number	Sets format to a custom number format that you specify.
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**), 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.



Books
Online



Find

Find
Again



Help



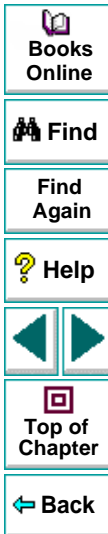
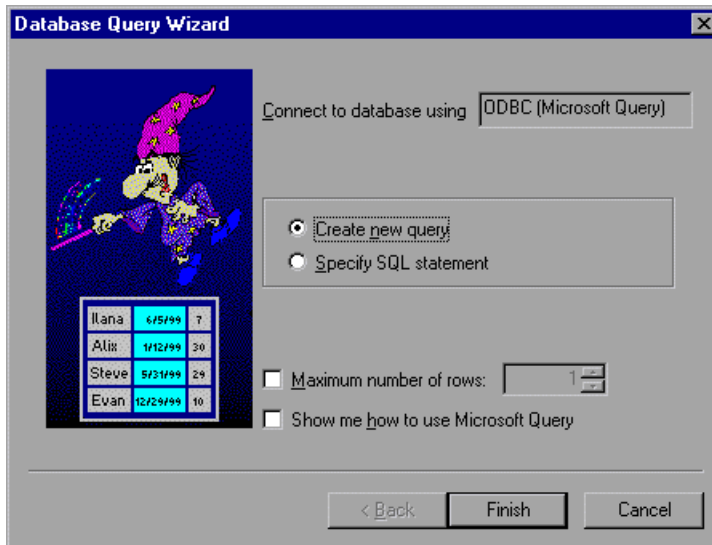
Top of
Chapter



Back

To import data from a database:

- 1 Right-click on the data table sheet to which you want to import the data and select **Data > Import from > ODBC Data Source**. The Database Query Wizard dialog box opens.



- 2 Select your database selection preferences and click **Next** or **Finish** (depending on which query option you selected). You can choose from the following options:
- **Create new query:** Opens Microsoft Query, enabling you to create a new query. Once you finish defining your query, you are exit back to Astra QuickTest.
 - **Specify SQL statement:** 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. If this check box is cleared, there is no maximum.
 - **Show me how to use Microsoft Query:** Displays an instruction screen before opening Microsoft Query when you click **Finish**. (Enabled only when **Create new query** is selected).

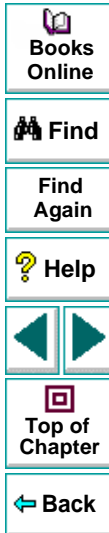


- 3 If you chose **Create new query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information about creating a query, see [Creating a Query in Microsoft Query](#) on page 322.

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



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



- **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:** Enter the SQL statement.
- 4 Astra QuickTest takes several seconds to capture the database query and restore the Astra 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. Astra 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 Astra QuickTest** to close Microsoft Query and return to Astra QuickTest.
 - Version 8.00 and Version 2000: In the Finish screen of the Query Wizard, select **Exit and return to Astra 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 Astra QuickTest** to close Microsoft Query and return to Astra QuickTest.
- 4 Return to step 4 on [page 321](#) to continue creating your database checkpoint in Astra 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 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.

Using Formulas to Create Input Parameterization Data

You can enter formulas rather than fixed values in the cells of an input 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.

	Date
1	3/28/2000
2	3/29/2000
3	3/30/2000
4	3/31/2000
5	4/1/2000
6	4/2/2000
7	4/3/2000



Books Online



Find

Find Again



Help





Top of Chapter



Back

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, Astra 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.


A1	=\$B1=337	
	Total Price	Total Price out
1	TRUE	337
2		

A FALSE result in the checkpoint column during a test run causes the test to fail.

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 5, [Creating Checkpoints](#).
- 2 Click **Parameter** and specify a logical name for the parameter.
- 3 Select the **Use data table formula** check box.
- 4 Specify your other checkpoint setting preferences as described in Chapter 5, [Creating Checkpoints](#).
- 5 Click **OK**.
- 6 Confirm the addition of two columns in the data table. The two columns are added to the table, and a checkpoint  icon is added to your test tree.
- 7 Highlight the value in the first (formula) column to view the formula and modify the formula to fit your needs.
- 8 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.



Using Data Table Scripting Functions

Astra QuickTest provides several data table functions 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](#).

For more details on the data table functions, refer to the *Astra QuickTest Function Reference*.

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 functions and properties that you can use to retrieve or set values.

The functions and properties available for each type of object are described below.



The DataTable Object

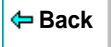
The table below summarizes the functions and properties of the DataTable object. For these functions and properties, use the syntax:

DataTable.PropOrFunc (*params*)

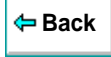
For example, the following statement returns the MySheet sheet.

DataTable.GetSheet ("MySheet")

Function or Property	Description
DataTable (<i>ParameterID</i> , <i>SheetID</i>) or DataTable.Value (<i>ParameterID</i> , <i>SheetID</i>)	Retrieves the value of the cell in the specified parameter and the current row.
DataTable (<i>ParameterID</i> , <i>SheetID</i>)= <i>newvalue</i> or DataTable.Value (<i>ParameterID</i> , <i>SheetID</i>)= <i>newvalue</i>	Sets the value of the cell in the specified parameter and the current row.



Function or Property	Description
RawValue (<i>ParameterID</i> , <i>SheetID</i>)	Retrieves the <i>raw value</i> of the specified parameter and current row. The <i>raw value</i> is the actual string written in the cell before it has been computed, such as the actual text of a formula.
GetSheetCount	Returns the total number of sheets in the data table.
GetSheet (<i>SheetID</i>)	Returns the specified sheet.
GlobalSheet	Returns the Global sheet.
LocalSheet	Returns the current local sheet.
AddSheet (<i>SheetName</i>)	Adds the specified sheet and returns it so that you can directly set or return properties of the new sheet in the same statement.
DeleteSheet (<i>SheetID</i>)	Deletes the specified sheet. Note that deleting the sheet will cause the test to fail if the corresponding action has parameterized values.
GetRowCount	Returns the total number of rows in the longest column of the Global sheet.



Function or Property	Description
GetCurrentRow	Returns the current row in the Global sheet.
SetCurrentRow (<i>RowNumber</i>)	Sets the specified row as the current row in the Global sheet.
SetNextRow	Sets the row after the currently active row as the new current row in the Global sheet.
SetPrevRow	Sets the row above the currently active row as the new current row in the Global sheet.
Import (<i>FileName</i>)	Imports the specified Excel file. Note that the imported table completely replaces all data in the existing data table.
Export (<i>FileName</i>)	Saves a copy of the data table in the specified location as an excel (.xls) file.



The Sheet Object

The table below summarizes the functions and properties of the Sheet object. For these functions and properties, use the syntax:

...Sheet.PropOrFunc (*params*).

For example, the following statement returns the name that Astra QuickTest assigned to the newly added Sheet, which may be different than the name specified, if the specified name already exists in the data table or if the name contains an invalid character.


DataTable.AddSheet ("MySheet").Name

In the above example, if the sheet, 'MySheet' already exists, Astra QuickTest returns *MySheet1* as the name of the new sheet.

Function or Property	Description
GetParameterCount	Returns the total number of parameters (columns) in the sheet.
GetParameter (<i>ParameterID</i>)	Returns the specified parameter from the sheet.
Name	Returns the name of the sheet.



Function or Property	Description
AddParameter (<i>ParameterName</i> , <i>value</i>)	Adds the specified parameter to the sheet, sets the value of the first row to the specified value, and returns the parameter so that you can directly set or return properties of the new parameter in the same statement.
DeleteParameter (<i>ParameterID</i>)	Deletes the specified parameter (column) from the sheet. Note that deleting a parameter from the data sheet will cause the test to fail if a corresponding parameter exists in the test.
GetRowCount	Returns the total number of rows in the longest column of the sheet.
GetCurrentRow	Returns the current row in the sheet.
SetCurrentRow (<i>RowNumber</i>)	Sets the specified row as the current row in the sheet.
SetNextRow	Sets the row after the currently active row as the new current row in the sheet.
SetPrevRow	Sets the row above the currently active row as the new current row in the sheet.




Books Online



Find

Find Again



Help





Top of Chapter



Back

The Parameter Object

The table below summarizes the functions and properties of the Parameter (column) object. For these functions and properties, use the syntax:

...**Parameter.PropOrFunc** (*params*)

For example, the following statement returns the name that Astra QuickTest assigned to the newly added Parameter (column), which may be different than the name specified, if the specified parameter name already exists in the sheet or if the name contains an invalid character.

DataTable.GetSheet("MySheet").AddParameter("ParamA", 2).Name

Function or Property	Description
Parameter or Parameter.value	Retrieves the value of the cell in the current row of the parameter.
Parameter=newvalue or Parameter.value=newvalue	Sets the value of the cell in the current row of the parameter.



Function or Property	Description
RawValue	Retrieves the <i>raw value</i> of the current row of the parameter. The <i>raw value</i> is the actual string written in the cell before it has been computed, such as the actual text of a formula.
ValueByRow (RowNum)	Returns the value of the cell in the specified row of the parameter.
Name	Returns the name of the parameter.



Creating Tests

Handling Unexpected Events and Errors

You can instruct Astra 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**

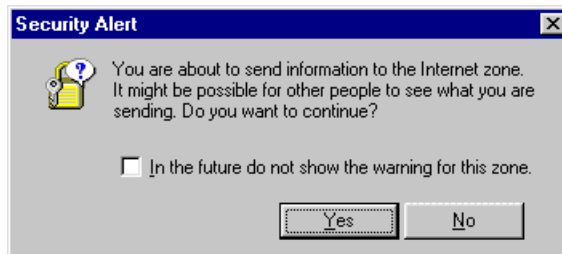


About Handling Unexpected Events and Errors

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 Astra 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, you can instruct Astra 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 Astra QuickTest supports. Each exception is associated with a handler function that is activated when there is a need to recover the test run. You can modify the list of exceptions and configure additional types of dialog box exceptions that you would like Astra QuickTest to support.

Note: If you do not want to associate a handler function with a dialog box, you can choose to always bypass the step containing the dialog box using an optional step. For additional information, see Chapter 17, [Running Tests](#).

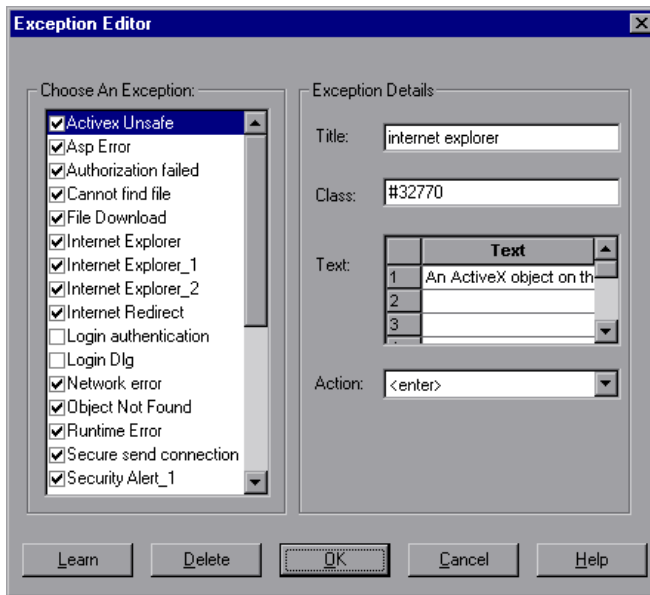


Changing the Status of Exceptions

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 appears 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.

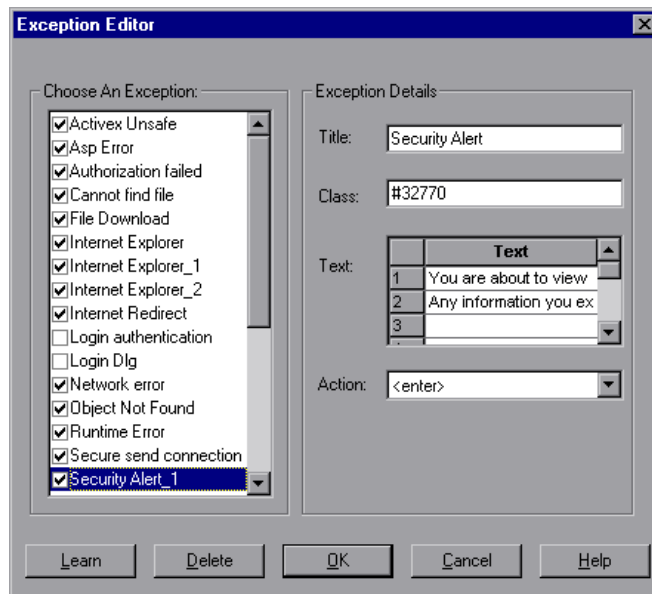


Modifying Exceptions

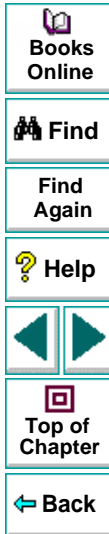
You can modify the details of an exception listed in the Exception Editor.

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 appears 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 appears in the exception dialog box, edit a text line in the **Text** box.
 - To change the handler function associated with this exception dialog, choose a function from the **Action** list. This function recovers the test run.

Function	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.

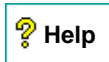


Adding New Exceptions

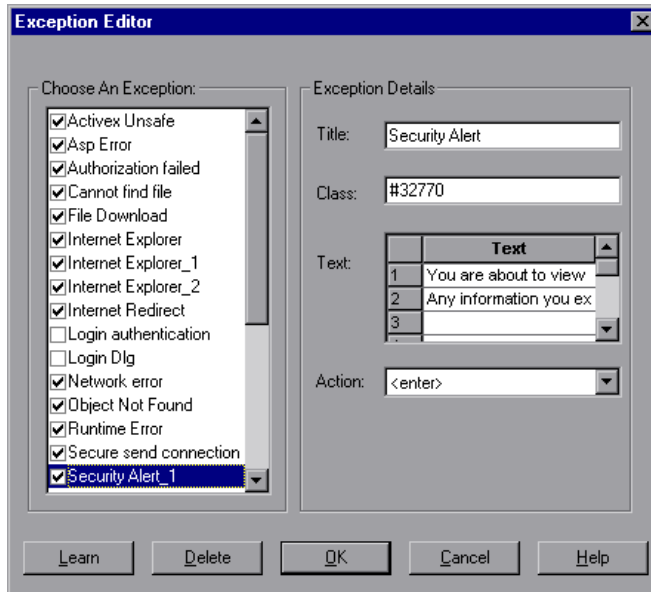
You can add a new exception to the list of exceptions in the Exception Editor.

To add a new exception:

- 1 Choose **Tools > Exception Editor**. The Exception Editor opens.



- Click the **Learn** button. The mouse pointer becomes a pointing hand. Click the dialog box. A new exception is added to the list.



The Exception Editor displays the title of the exception dialog box, the property class of the exception, the text that appears in the dialog box, and the handler function that is responsible for recovering test execution. To modify these fields, refer to [Modifying Exceptions](#) on page 340.

- 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 appears 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.



Running and Debugging Tests



Running and Debugging Tests

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](#)
- [Using Optional Steps](#)
- [Running a Test Batch](#)



About Running Tests

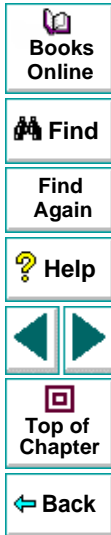
When you run a test, Astra QuickTest navigates through your Web site, performing the steps you recorded. If your test does not contain parameterized values, Astra QuickTest runs the test once. If the test contains global parameters (in the Global tab of the Data pane), Astra QuickTest runs the test for each row in the table, using the parameters you specified.

If the test contains parameters, you can specify whether to run the test, or a specific action in iterations, or to run the test or action in iterations for a range of data sets. For additional information, see Chapter 25, [Setting Testing Options for a Single Test](#) and Chapter 14, [Working with Actions](#).

Once the test run is complete, Astra QuickTest displays a report detailing the test results.

You can set up a batch of tests and run them sequentially using the Astra QuickTest Test Batch Runner. For more information see [Running a Test Batch](#) on page 355.

You can also run tests on objects with dynamic descriptions. For additional information, see Chapter 4, [Understanding How Astra QuickTest Identifies Objects](#).



Running a Test

When you run a test, Astra QuickTest performs the steps you recorded on your Web site and displays each Web page in your browser. Astra QuickTest always runs a test from the first step in the test.

If your test does not contain parameterized values, Astra QuickTest runs the test once. If the test does contain parameters, Astra 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](#).

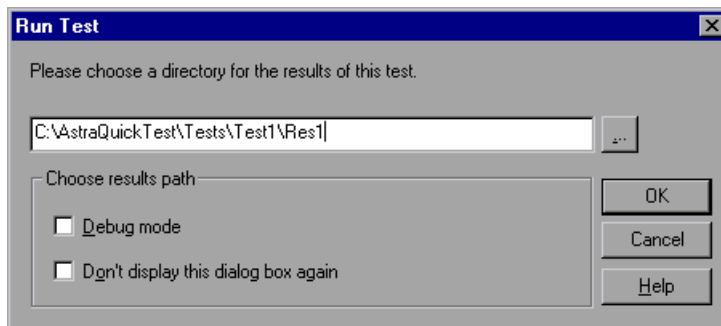


To run a test:

- 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 Test dialog box opens, displaying the path and a default test run name for the test results.



- 3 To save the test 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 results, select the **Debug mode** check box.

Note: You cannot change the test results name when **Debug mode** is selected.



If you do not want to display the Run Test dialog box the next time you run your test, select the **Don't show again** check box.

- 4 Click **OK**. The Run Test dialog box closes and Astra QuickTest begins running the test. Astra QuickTest always runs a test from the first step in the test. As Astra 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** option in the Options dialog box. For more information about the Options dialog box, see Chapter 24, [Setting Astra QuickTest 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 pauses. To resume running a paused test, click the **Run** button or choose **Test > Run**.



Click the **Stop** button or choose **Test > Stop**. The test stops running and the **Test Results** window opens.



Using Optional Steps

When running a test, if a step does not open a particular dialog box, Astra QuickTest does not necessarily interrupt the test run. It can bypass the step and continue to run the test. Astra QuickTest will bypass any *optional* step. By default, Astra 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](#).




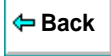
Setting Optional Steps

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 fails to open an optional dialog box, Astra 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.

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 *Astra QuickTest Function Reference*.

Default Optional Steps

Astra 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



Logical Name	Title
Runtime error	Error
Security Alert	Security Alert
Security Information	Security Information
Security Warning	Security Warning
Username and Password Required	Username and Password Required



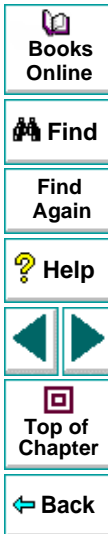
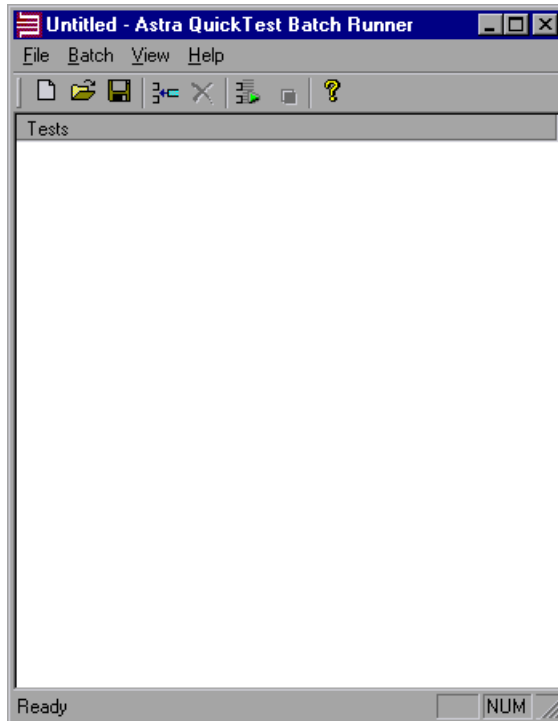
Running a Test Batch

You can use the Astra QuickTest Test Batch Runner to run several tests in succession. The results for each test are stored in their default location. Using the Astra QuickTest 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.



To set up and run a test batch:

- 1 Choose **Programs > Astra QuickTest > Tools > Test Batch Runner** from the Start menu. The Astra QuickTest Batch Runner opens.





- 2 Click the **Add** button or choose **Batch > Add**. The Open Astra 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 Astra 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](#).



Running and Debugging Tests

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 a Checkpoint**
- **Viewing the Runtime Data Table for a Parameterized Test**
- **Printing Test Results**



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 362.

The screenshot shows the Mercury Test Results window for a test named 'MercTours1'. The window is titled 'MercTours1 [Res1] - Mercury Test Results'. It features a menu bar (File, View, Help) and a toolbar with icons for file operations and help. On the left, there is a 'Test results tree' showing a hierarchy: 'Test MercTours1 Summary' (with a red X icon), 'Runtime Data' (with a blue grid icon), 'MercTours1 Iteration 1 (Rc)' (with a green checkmark icon), and 'MercTours1 Iteration 2 (Rc)' (with a red X icon). The main area displays the 'MercTours1 Results Summary'. It includes the test name, execution start and end times, and a table of iteration results. Below this is a 'Statistics' section with another table showing the count of passed, failed, and warning results.

Test results tree —

Test results details —

MercTours1 Results Summary

Test : MercTours1

Execution started : 8/20/00 - 11:39:06

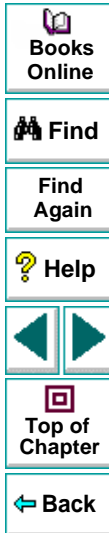
Execution ended : 8/20/00 - 11:40:00

Iteration #	Results
1	Passed
2	Failed





Statistics :

Status	Times
Passed	1
Failed	1
Warnings	0

For Help, press F1



Test Results Tree

The left pane in the Test Results window displays the *test results tree*—a graphical representation of the test results. This includes the  icon for a successful iteration, the  icon for a failed iteration, and the  icon for warnings. In the example above, the tree includes two 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 values retrieved from a parameterized test while it runs (output parameters). 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 appears. 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 463.



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**. The Select Results File dialog box opens. Select a results file (.qtp extension). Click **Open**. The **Test Results** window opens.

The screenshot shows the Mercury Test Results window for 'MercTours1'. The window title is 'MercTours1 [Res1] - Mercury Test Results'. The interface includes a menu bar (File, View, Help) and a toolbar. On the left, there is a 'Test results tree' showing a hierarchy: 'Test MercTours1 Summary', 'Runtime Data', 'MercTours1 Iteration 1 (Rc)', and 'MercTours1 Iteration 2 (Rc)'. The main area displays 'Test results details' for 'MercTours1 Results Summary'. It shows the test name, execution start and end times, and a table of iteration results. Below that, it shows statistics for the test run.

Test results tree

Test results details

MercTours1 Results Summary

Test : MercTours1

Execution started : 8/20/00 - 11:39:06

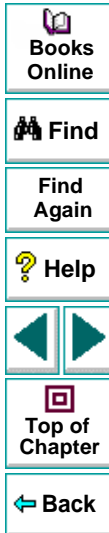
Execution ended : 8/20/00 - 11:40:00

Iteration #	Results
1	Passed
2	Failed

Statistics :

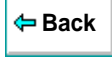
Status	Times
Passed	1
Failed	1
Warnings	0

For Help, press F1



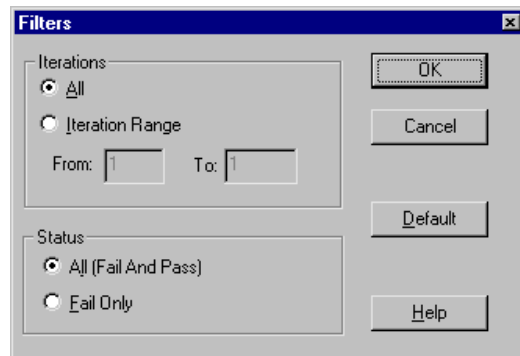
- 2 You can collapse or expand a branch in the test results tree in order to change the level of detail that the tree displays.
 - To collapse a branch, click the Collapse (-) sign to the left of the branch icon. The report tree hides the details for the branch and the Collapse sign changes to Expand.
 - 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: The 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 **Filter** button or choose **View > Filters**. The Filters dialog box opens.

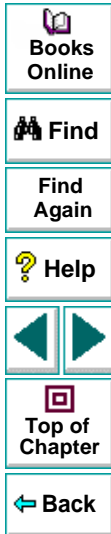


The default filter options are displayed above.

- 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**. The Open dialog box opens. Return to the root folder of the test. Select a Results folder (i.e. Res3), and then select the Report folder. Select the results file (.qtp extension) and click the **Open** button.





6 To print the test results, see [Printing Test Results](#) on page 371.

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 > Report Viewer**.



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, Astra 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 checkpoint, see Chapter 5, [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**. The Select Results File dialog box opens. Select a results file (*.qtp* extension). Click **Open**. The Test Results window opens.
- 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.

The screenshot shows the Mercury Test Results window for a test named 'MercTours1 [Res4]'. The left pane displays a tree view of the test execution, with the 'Checkpoint: "depart"' node selected. The right pane shows the 'Details' for this checkpoint, indicating it has failed. Below the title, a table titled 'depart Results' compares the expected and actual values for the 'depart' action.

Property Name	Property Value
value	Seattle
value	Acapulco
type	select-one
name	depart
items count	10

Below the table, a blue banner displays the text: "... nothing but Blue from now on ...".



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 “Seattle”, but the actual value is “Acapulco”.

- 4 Choose **File > Exit** to close the Test Results window.

Viewing the Runtime Data Table for a Parameterized Test

After you run a parameterized test, the Runtime data table displays the values used to run a parameterized test, or the values retrieved from a parameterized test while it runs (output parameters). For more information on parametrization, see Chapter 11, [Parameterizing Tests](#). For more information on output parameterization, see Chapter 12, [Creating Output Parameters](#). 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**. The Select Results File dialog box opens. Select a results file (*.qtp* extension). Click **Open**. The Test Results window opens.





- In the left pane of the Test Results window, click the **Runtime Data** icon. The right pane displays the Runtime data table.

The screenshot shows the Mercury Test Results window titled "Sample [Res3] - Mercury Test Results". The left pane shows a tree view with "Test Test1 Summary" expanded, and "Runtime Data" selected. Below it are "Test Iteration 1" (passed), "Test Iteration 2" (failed), and "Test Iteration 3" (failed). The right pane displays a table with the following data:

	depart value	arrive value	C	D	E
1	Frankfurt	Portland			
2	Portland	Acapulco			
3	Seattle	London			
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					

The status bar at the bottom indicates "Global Action1 /".

- Books Online
- Find
- Find Again
- Help
-
- Top of Chapter
- Back

In the above example, the Runtime data table contains the parameterized flight departure values and the flight arrival values.

- Choose **File > Exit** to close the Test Results window.

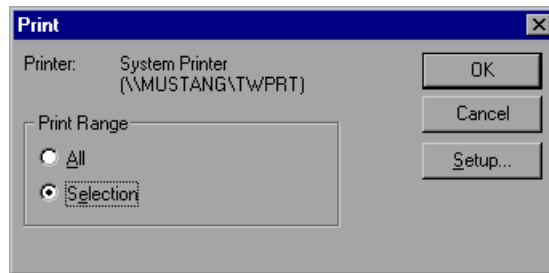
Printing Test Results

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.



Running and Debugging Tests

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.

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 Astra 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 Astra QuickTest window.



Books
Online



Find



Find
Again



Help



Top of
Chapter



Back

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.

A small icon of an open book.

**Books
Online**

A small icon of a magnifying glass over a document.

Find

**Find
Again**

A small yellow question mark icon.

Help

A small icon of a square with a smaller square inside, representing a home or top button.

**Top of
Chapter**

A small blue left-pointing arrow icon.

Back

Setting Breakpoints

By setting a breakpoint you can stop a test run at a specific place in a test. A breakpoint is indicated by a red-colored hand in the left margin of the test window. Astra 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 appears in the left margin of the Astra QuickTest window.



Note: The breakpoints you define are active only during your current Astra QuickTest session. If you terminate your Astra 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 Astra 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 Astra QuickTest window.

Using the Debugger Views

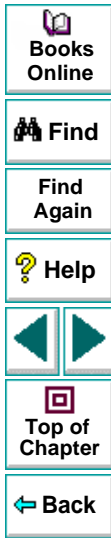
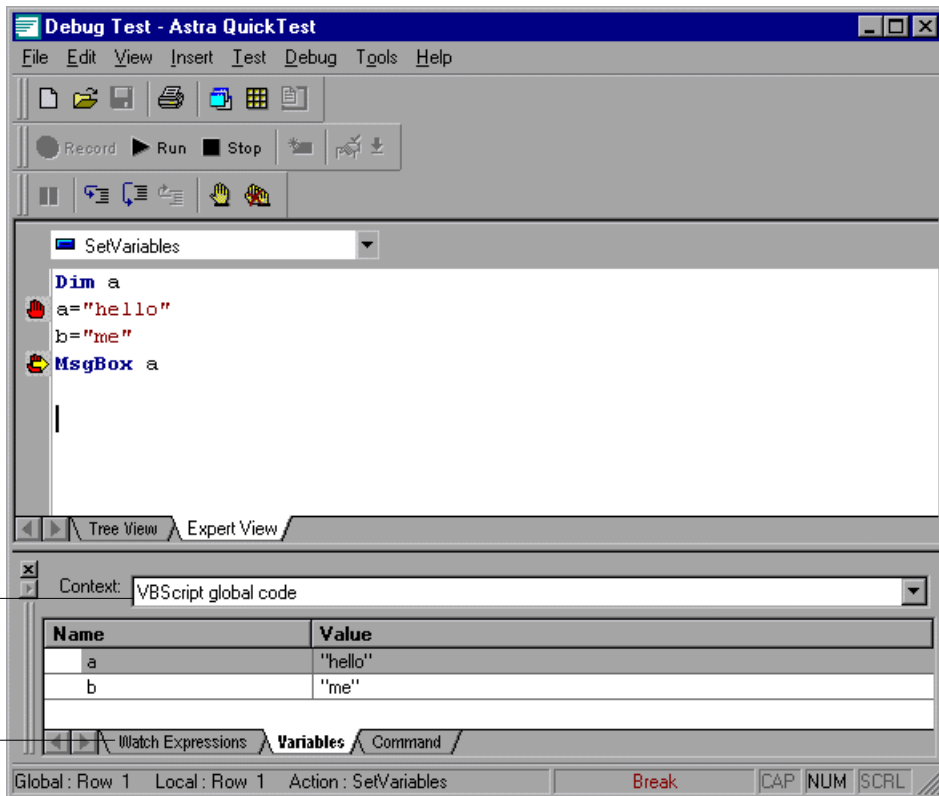
When a test stops at a breakpoint, you can use the Debugger pane to view, set, modify the current value of objects or variables in your test.

To open the Debugger pane:

- 1 Run a test with one or more breakpoints.



- When the test pauses at the first breakpoint, choose View > Debugger Views. The Debugger pane opens at the bottom of the Astra 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 Astra 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.

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.



Books
Online



Find

Find
Again



Help



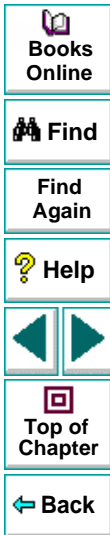
Top of
Chapter



Back

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, Astra 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](#).



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Step 2: Add Breakpoints

Add a breakpoint at line 2 and line 4. For more information about adding breakpoints, see [Setting Breakpoints](#) on page 376.

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 > Debugger Views** 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 **Empty**, because the breakpoint stopped after the variable a was declared, but before the value of 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.

Select the **Variables** tab in the Debugger pane. Note that the variable a is displayed with the value **Empty**, because a is the only variable that has been declared at this point in the test.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back



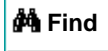
Step 5: Check the Value of the Variables at the Next Breakpoint

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.



Advanced Features

 Books
Online

 Find

Find
Again

 Help



 Top of
Chapter

 Back

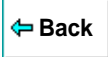
Advanced Features

Configuring Event Recording

If Astra 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**
- **Resetting Standard Event Recording Configuration Settings**



About Configuring Event Recording

Astra QuickTest records your test by recording the *events* you perform on your Web site. 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 Astra QuickTest automatically records by default. You can modify the default event recording settings by using the Event Configuration Settings 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, Astra 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 that event configuration is a global setting and therefore affects all tests that are recorded after you change the settings.

Note: Changing the event configuration settings does not affect tests that have already been recorded. If you find that Astra 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, Astra QuickTest uses the *Basic* recording configuration level. If Astra QuickTest does not record all the events you need, you may require a higher event configuration level.

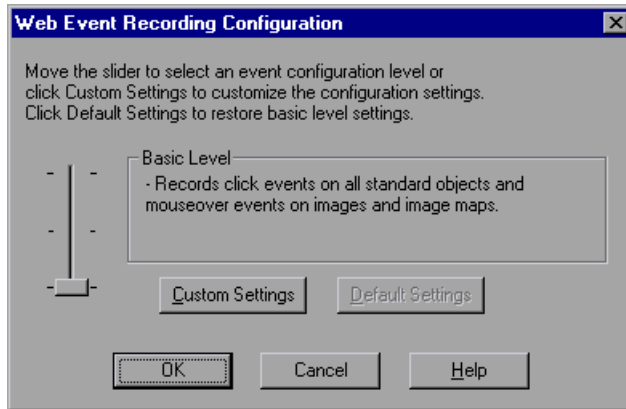
The Event Configuration Settings dialog box offers three standard event configuration levels.

Level	Description
Basic	<p>Default</p> <ul style="list-style-type: none"> - Always records click events on standard Web objects. - Always records reset and submit events 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.
Medium	Records click events on the <DIV>, , 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 <i>handlers</i> or <i>behaviors</i> attached in addition to the objects recorded in the basic level. For more information on handlers and behaviors, see Listening Criteria on page 397.



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 Event Configuration dialog box.

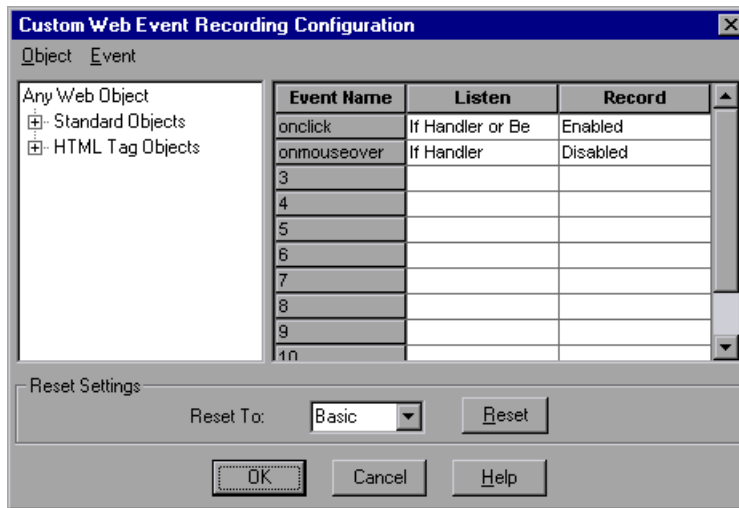
The Custom Event Configuration dialog box enables you to customize event recording in several ways. You can:

- add or delete objects to which Astra QuickTest should apply special listening or recording settings
- add or delete events for which Astra QuickTest should listen for all objects
- add or delete events for which Astra QuickTest should listen for one or more specific objects
- modify the listening or recording settings of an event for which Astra 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:

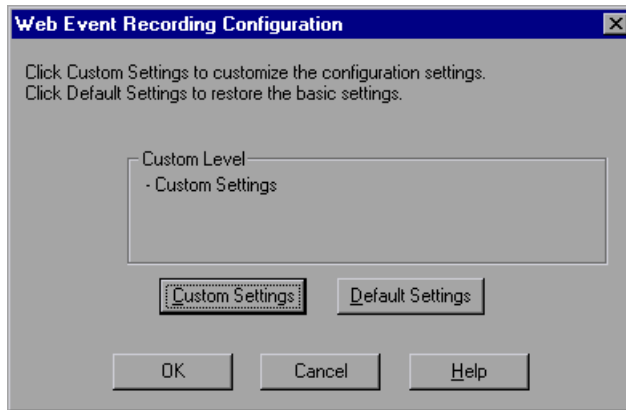
- 1 Choose **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
- 2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.



- 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 Event 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.

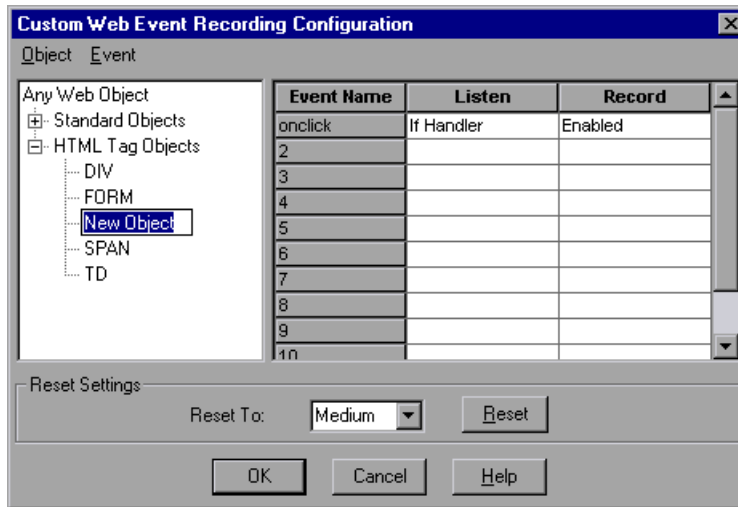
When working with the objects in the Custom Web Event Recording Configuration dialog box, keep the following principles in mind:

- 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.
- Astra 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 appears in the HTML Tag Objects list.

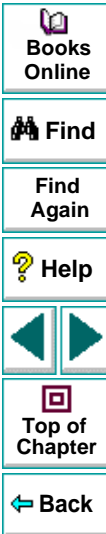


- 2 Click **New Object** to rename it. Enter the exact HTML Tag name.

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 395.

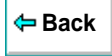
For more information on listening and recording settings, see [Modifying the Listening and Recording Settings for an Event](#) on page 397.



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.

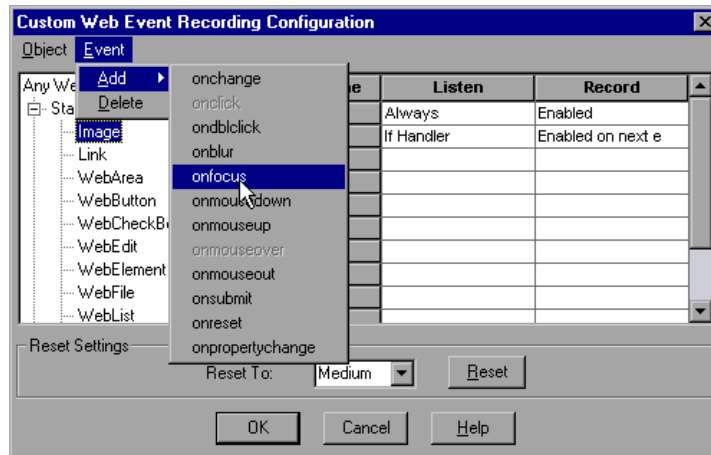


Adding and Deleting Listening Events for an Object

You can modify the list of events that trigger Astra 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 appears 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 397.



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 Astra 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**.



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

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

You can select **Always**, **If Handler**, **If Behavior**, or **If Handler or Behavior**.



Books
Online



Find



Find
Again



Help



Top of
Chapter



Back

Recording Status

For each event to which Astra 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 appears 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**.



- In the row of the event you want to modify, select the recording status you want from the Record column.

Event Name	Listen	Record
onclick	Always	Enabled
onmouseover	If Handler	Enabled on next
3		Disabled
4		Enabled
5		Enabled on next ev
6		
7		
8		
9		
10		



Resetting Standard Event Recording Configuration Settings

If you want to restore standard settings after you have set Custom settings, there are two ways to reset the settings.

- You can reset the event recording configuration settings to the basic level from the Web Event Recording Configuration dialog box.
- You can reset the settings to any one of the standard settings from within the Custom Web Event Recording Configuration dialog box so that you can begin customizing from that point.

Note: When you choose to reset standard settings, your custom settings are cleared completely.

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 re-appears 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 387.



Books
Online



Find

Find
Again



Help



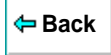
Top of
Chapter



Back

To reset standard settings 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.



Advanced Features

Enhancing Your Tests with Programming

After recording a test, you can use Astra QuickTest to enhance your test using a few simple programming techniques.

This chapter describes:

- **Inserting Functions**
- **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, Astra 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. Astra QuickTest includes the *Function wizard*, a programming tool that helps you to quickly and easily add recordable and non-recordable functions to your test. You can use the wizard to add functions 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 function.

Astra 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 Astra 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 Functions

After recording, you can add additional functions to your tests using the Function wizard. With the wizard you can add recordable and non-recordable functions that perform operations on objects or retrieve information from your site. For example, the **QueryValue** function enables you to query the method argument value. You can use the return value of the function as an output parameter or as part of a conditional statement.

To insert a function in a test:

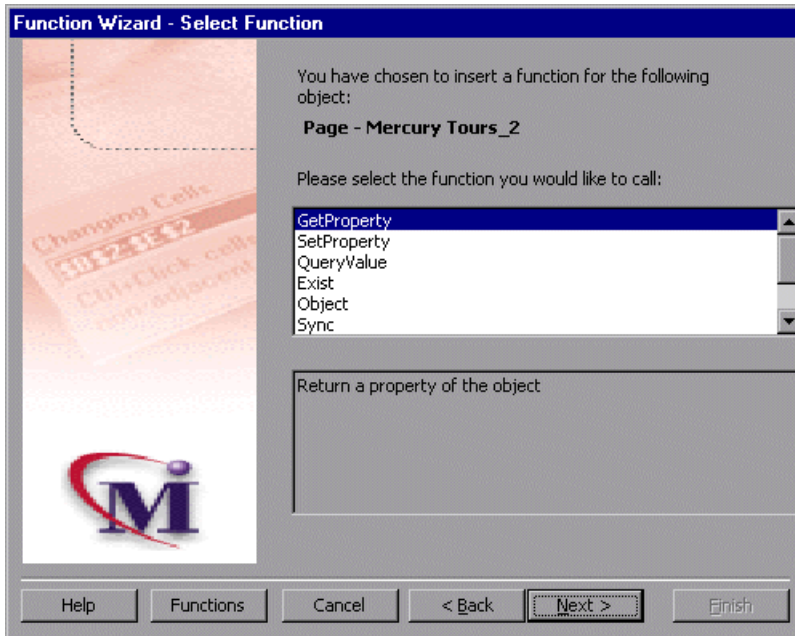
- 1 In the Tree View, right-click a step in the test tree and choose **Insert > Step > Function**.

Tip: To add a function from the Expert View, click a statement in the test script. Right-click the highlighted object in the ActiveScreen and choose **Insert Function**. The **Object Selection - Insert Function** opens. Select an object and click **OK**.

- 2 The Function Wizard - Introduction screen opens. Click **Next**.

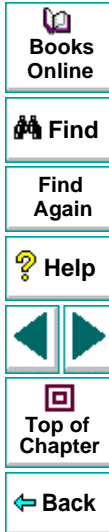
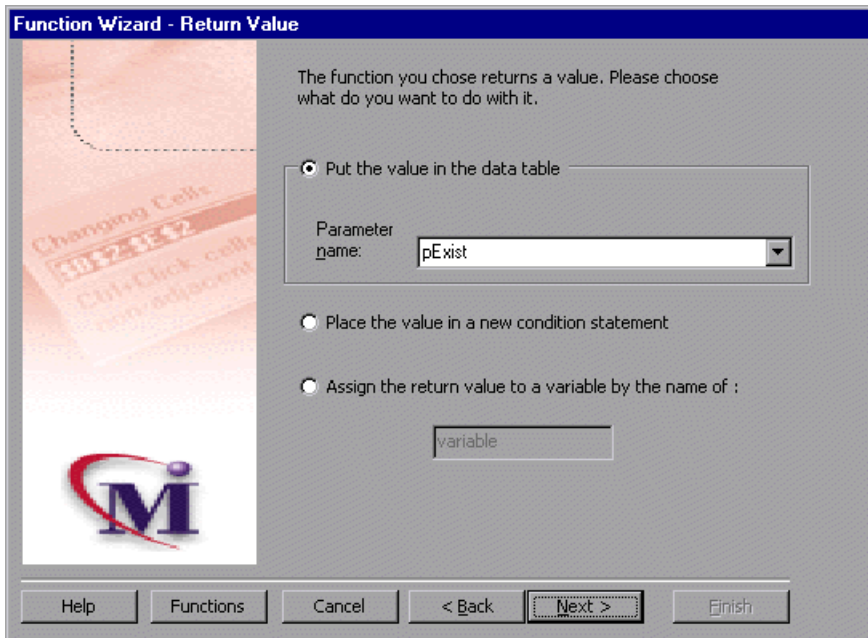


3 The Function Wizard - Select Function screen opens.



Select a function and click **Next**.

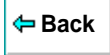
- 4 If the function you chose returns a value, the Function Wizard - Return Value screen opens. Otherwise, proceed to the next step.



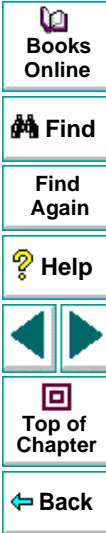
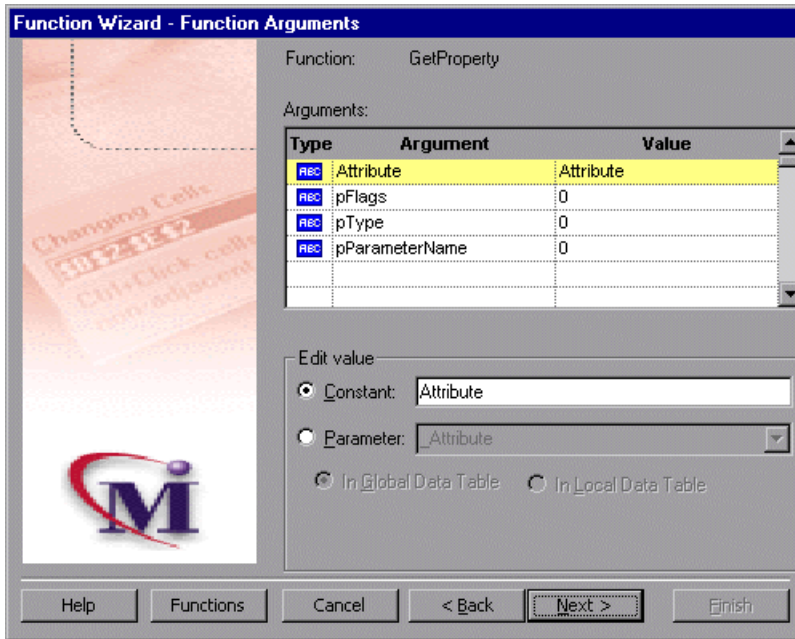
The following options are available:

Option	Description
Put the value in the data table (default)	Inserts the return value of the function as an output parameter into your Data pane. For more information, see Chapter 12, Creating Output Parameters .
Parameter name	Sets the name for the output parameter. You can accept the default name, select from the list, or enter a new name. For more information, see Chapter 12, Creating Output Parameters .
Place the value in a new condition statement	Inserts the return value of the function into a conditional statement. For more information, see Using Conditional Statements on page 412.
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.



- If the function you chose has function arguments, the Function Wizard - Function Arguments screen opens. Otherwise, proceed to the next step.



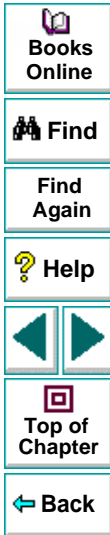
The screen displays the arguments you can parameterize, in a pane listing the arguments, their values, and their types:

Option	Description
Type	The  icon indicates that the argument value is a constant. The  icon indicates that the argument value is a parameter.
Argument	The name of the argument whose value will be parameterized.
Value	The value of the argument to parameterize.

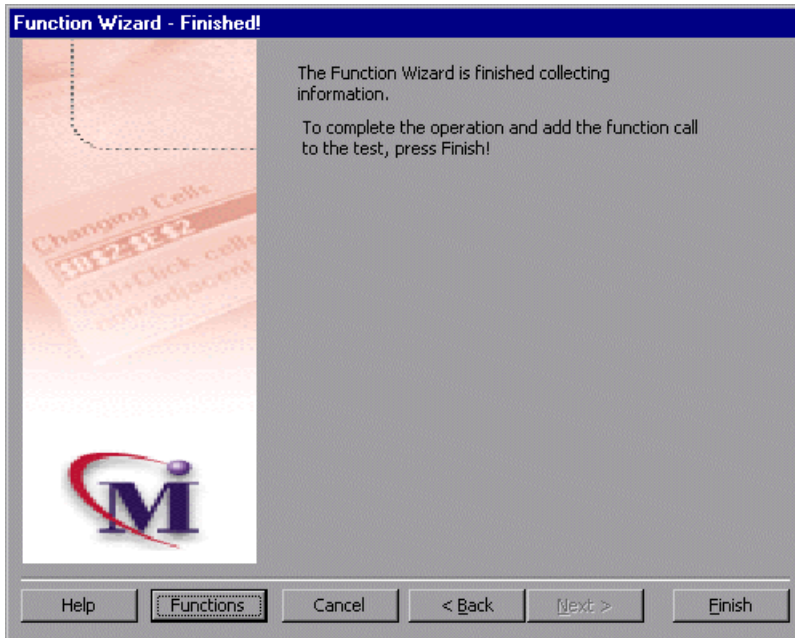
In the **Edit value** section, you use the following options to edit the argument value.

Option	Description
Constant (default)	Sets the argument value as a constant.
Parameter	Sets the argument value as a parameter. For more information, see Chapter 11, Parameterizing Tests .

Click **Next** to continue.



- 6 The Function Wizard - Finished screen opens.



Click **Finish** to complete the process and add the function to your test.

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* =.

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
```



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

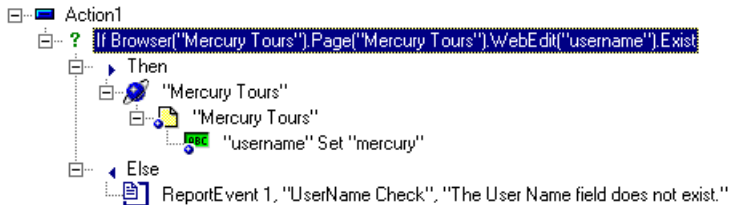
Part of Statement	Description
condition	One or more expressions that evaluate to true or false. If the condition is null, it is treated as false.
statements	One or more statements, separated by colons, that are executed if the condition is true.
condition-n	Same as <i>condition</i> .
elseif statements	One or more statements, separated by colons, that are executed if the associated <i>condition-n</i> is true.
else statements	One or more statements, separated by colons, that are executed if no previous <i>condition-n</i> expression is true.
End If statement	Terminates each If statement.

For example, the statement below (as it appears 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:



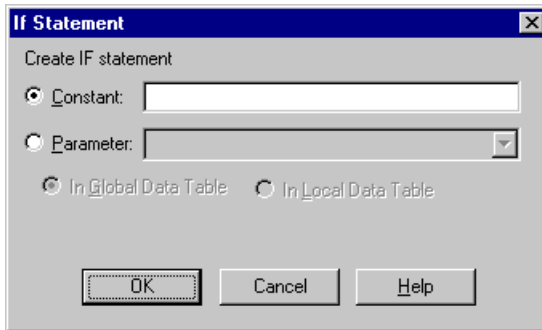
In the test tree, the following icons are used to indicate the different levels of *If...Then...Else* statements:

Icon	Description
	Starts an <i>If</i> statement.
	Starts a <i>Then</i> statement.
	Starts an <i>Elseif</i> statement.
	Starts an <i>Else</i> statement.

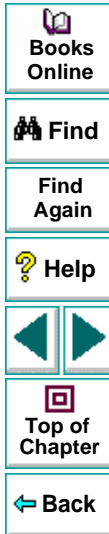


To add a conditional statement:

- 1 In the Tree View, click a step in the test tree.
- 2 Choose **Insert > Step > If...Then...Else > 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 a parameter from the list or enter a new parameter. For example, type $a > c$ for a formula in a table. For more information, see Chapter 11, [Parameterizing Tests](#).
 - To add the parameter to the Global tab in the Data pane, select **Global**. To add the parameter to the Action tab, select **Local**. For more information, see Chapter 11, [Parameterizing Tests](#).



- 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 > If...Then...Else > If...Then
an Elseif statement	Insert > Step > If...Then...Else > Elseif...Then
an Else statement	Insert > Step > If...Then...Else > 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.

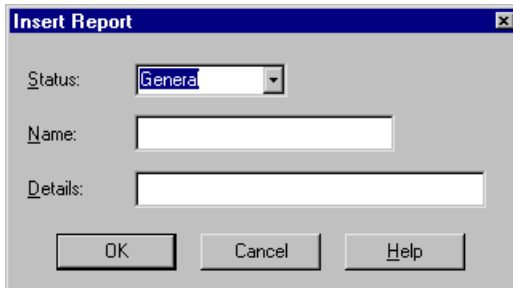


Sending Messages to Your Test Results

You can define a message in your test that Astra 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, Astra 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, right-click a step and choose **Insert > Step > Report**. The Insert Report dialog box opens.




- 2 Select the status that will result from this step from the **Status** list.

Status	Description
Passed	Causes this step of the test to pass. Sends the specified message to the report.
Failed	Causes this step of the test (and therefore test) to fail. Sends the specified message to the report.
General	Sends a message to the report without affecting 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.

After you run the test, the  icon in the Test Results window indicates that the message was sent.

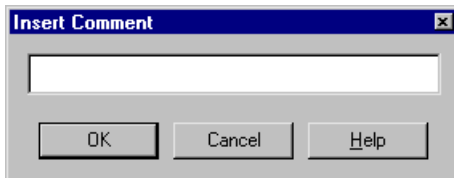


Adding Comments


While programming, you can add comments to your tests. A *comment* is an explanatory remark in a program. When you run a test, Astra 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 as *Rem.*



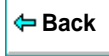
Advanced Features

Testing in the Expert View

In Astra 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**



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.

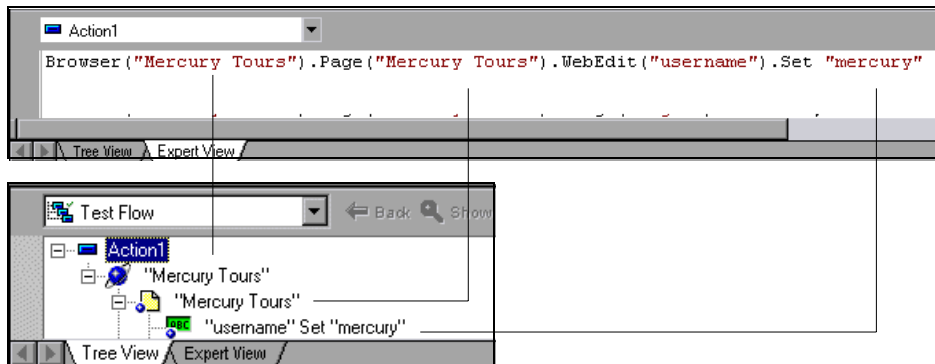
In the Expert View you can also add functions manually instead of from the Function wizard. For information on using the Function wizard, see Chapter 21, [Enhancing Your Tests with Programming](#).



Understanding the Expert View

Astra QuickTest can display tests you record in two formats:

- the Tree View, where Astra QuickTest displays the object hierarchy in an icon-based tree
- the Expert View, where Astra QuickTest displays the object hierarchy in VBScript






Note that in the diagram above, the object hierarchy is identical in both views.

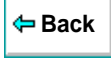
Each line of VBScript in the Expert View represents a step in the test. The example above represents a step in the test in which the user inserts the name "mercury" into an edit field. The hierarchy of the step enables you to see the name of the site, the name of the page, the name of the object in the page, and the name of the method performed on the object.



When you record your test, Astra QuickTest records the operations you perform on your Web site in terms of the objects on the page. It identifies the objects on a Web page by specific names (e.g. a button or a list) and the function 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 function performed on the edit box is "Set". The name inserted into the edit box is "mercury".



An object's logical name appears 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 function performed on the object always appears 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** function:

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set  
"mercury"
```

For a complete list of objects and their associated functions, choose **Help > Astra QuickTest Function Reference** to open the *Astra QuickTest Function Reference*.



Checkpoints

In Astra QuickTest, you create checkpoints on pages, text strings, objects, and tables. When you create a checkpoint in the Tree View, Astra QuickTest creates a corresponding line in VBScript in the Expert View. It uses the **Check** function to perform the checkpoint.

For example, in the following statement Astra QuickTest performs a check on the word “confirmed”:

```
Browser("Mercury Tours").Page("Flight Confirmation").  
    Check Checkpoint("confirmed")
```

The corresponding step in the Tree View appears as follows:

 Checkpoint "confirmed"

Note: You cannot insert checkpoints into your test while in the Expert View. Use the Tree View to insert and modify checkpoints. For more information on inserting and modifying checkpoints, see Chapter 5, **Creating Checkpoints**.



Parameters

You can use Astra 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, Astra QuickTest creates a corresponding line in VBScript in the Expert View.

Astra QuickTest calls the values of a parameterized object from the data table using the following syntax:

function_name DataTable (parameterID, sheetID)

<i>function_name</i>	The name of the function that Astra QuickTest executes on the parameterized object.
DataTable	The data table object.
<i>parameterID</i>	The name of the column in the data table.
<i>sheetID</i>	The name of the sheet. If the parameter is a global parameter, the word “global” appears.

Note: You cannot create parameters from the Expert View. Use the Tree View to create parameters. For more information on parameterization, see Chapter 11, [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 function 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 function.

Most objects have corresponding functions. For example, the **Back** function is associated with the Browser object. Some functions are associated with more than one object. For example, the **Click** function may be used on the WebArea object as well as the WebButton object.

The objects in Astra QuickTest are divided by environment.

Astra QuickTest environments include Web objects, ActiveX objects, DataTable objects, Utility objects, Java objects, Macromedia Flash objects, and Real Player objects.

Note: Support for Java objects, Macromedia Flash objects, and Real Player objects are supported in Astra QuickTest Professional only.



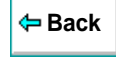
Functions Associated with Web Objects

The following table summarizes the Web objects and the functions with which they are associated.

Object	Description	Functions
Browser	the site	AddCookie, Back, Check, Close, Exist, Forward, FullScreen, GetProperty, Highlight, Home, Navigate, QueryValue, Set, SetProperty, Stop, Sync
Page	the page	Check, EndTransaction, Exist, GetProperty, Highlight, QueryValue, SetProperty, StartTransaction, Sync
Frame	the frame	Exist, GetProperty, QueryValue, SetProperty
Image	an image with or without a target URL	Check, Click, Exist, FireEvent, GetProperty, Highlight, MouseOver, Output, QueryValue, SetProperty
Link	a hypertext link	Check, Click, Exist, FireEvent, GetProperty, Highlight, MouseOver, Output, QueryValue, SetProperty



Object	Description	Functions
WebArea	a client-side image map	Check, Click, Exist, FireEvent, GetProperty, Highlight, MouseOver, QueryValue, SetProperty
WebButton	a button	Check, Click, Exist, FireEvent, GetProperty, Highlight, MouseOver, Output, QueryValue, SetProperty
WebCheckBox	a check box with "ON" and "OFF" states	Check, Click, Exist, FireEvent, GetProperty, Highlight, MouseOver, QueryValue, Set, SetProperty
WebEdit	an edit field	Check, Click, Exist, FireEvent, GetProperty, Highlight, MouseOver, QueryValue, Set, SetProperty, SetSecure, Submit
WebElement	A general Web object. This object is recorded for HTML tags such as DIV and SPAN that are not represented by a specific Web object.	Check, Click, Dblclick, Exist, FireEvent, GetProperty, MouseOver, QueryValue, SetProperty



Object	Description	Functions
WebFile	an edit field with a "Browse" button attached	FireEvent, GetProperty, Highlight, QueryValue, Set, SetProperty
WebList	a dropdown or multi-selection list	Check, Click, DeSelect, Exist, ExtendSelect, FireEvent, GetProperty, Highlight, MouseOver, QueryValue, Select, Set, SetProperty
WebRadioGroup	a group radio buttons with the same name.	Check, Click, Exist, FireEvent, GetProperty, Highlight, MouseOver, Output, QueryValue, Select, Set, SetProperty
WebTable	a table	CellText, CellTextByContext, Check, ChildItem, ChildItemCount, Click, ColumnCount, Exist, FireEvent, GetProperty, Highlight, MouseOver, Output, QueryValue, RowCount, RowText, RowTextByContext, SetProperty, TextCellExist, TextRowExist



Note: Objects and functions for other environments are described in the relevant chapters.

- For details on ActiveX objects and functions, see Chapter 8, [Testing ActiveX Controls](#).
- For details on data table objects and functions, see Chapter 15, [Working with Data Tables](#).
- For details on Java objects and functions, see Chapter 9, [Testing Java Applets](#).
- For details on Macromedia Flash objects and functions, see Chapter 10, [Testing Multimedia Applications](#).
- For details on Real Player objects and functions, see Chapter 10, [Testing Multimedia Applications](#).
- For details on Utility objects and functions, see [Functions Associated with Utility Objects](#) on page 434.



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** function 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:

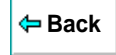
```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select
"Paris"
```

Browser object Mercury Tours

Page object Find Flights

WebList object (combo box or list box) depart

The **Select** function selects Paris in the WebList object.



Note: For the syntax for Astra functions, refer to the *Astra QuickTest Function Reference*. To open this, choose **Help > Astra QuickTest Function Reference**.

Functions Associated with Utility Objects

Utility objects and functions enable you to control how your test runs.

Setting the Run Mode

By default, Astra QuickTest runs the test by events. If it is necessary to run the test, or part of the test, directly by mouse operations, you can change this setting in the test script from the Expert view with the **WebPackage ("ReplayType")** function.

To set the test to run by mouse operations, enter:

```
Setting.WebPackage("ReplayType") = 2
```

To return to the event run mode, enter:

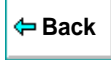
```
Setting.WebPackage("ReplayType") = 1
```

Note: The **WebPackage(ReplayType)** function applies only to Web objects.



The following table summarizes all Utility objects and the properties and functions with which they are associated:

Object	Description	Functions
Crypt	The object used to encrypt strings.	Encrypt
OptionalStep	object that causes the step to which it is attached to be bypassed, if the object referred to in the step cannot be found. A bypassed step is an optional step.	N/A
Reporter	the object used to send information to the test report	ReportEvent
Setting	the object used to modify test settings during the test run.	Add, AutomaticLinkRun, DefaultLoadTime, DefaultTimeOut, Remove, WebTimeOut
WebPackage	the object used to modify test settings specific to Web objects during the test run.	ReplayType



Functions Not Associated with Objects

In addition to the functions discussed above, there are functions that can be inserted into the test script that are not associated with any object.

Adding a Pause During a Test Run

In the following example, the user records a test on the Mercury Tours sample site, which selects an arrival city and a departure city. The following statements are recorded in the Expert View:

```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").
  Select"New York"
Browser("Mercury Tours").Page("Find Flights").WebList("arrive").Select
"Paris"
```

In order to instruct Astra QuickTest to pause between these two steps, the user inserts the **Wait** function between the steps. The **Wait** function has the following syntax:

Wait (*seconds*)

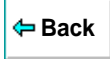
In order to pause the test for 10 seconds, the user inserts 10 as the value for the *seconds* argument. The script now appears as follows:

```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").
  Select "New York"
wait(10)
```



Browser("Mercury Tours").Page("Find Flights").WebList("arrive").Select
"Paris"

When the test runs, Astra QuickTest pauses for 10 seconds between selecting
the departure city and the arrival city.



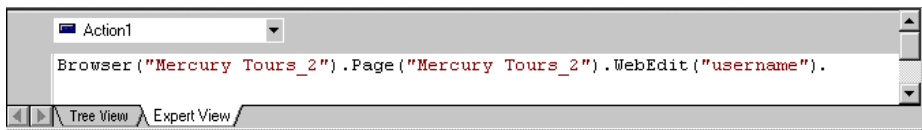
Generating a Function for an Object

In the Expert View you can generate functions. You can also generate functions in the Tree View using the Function wizard. For additional information, see Chapter 21, [Enhancing Your Tests with Programming](#).

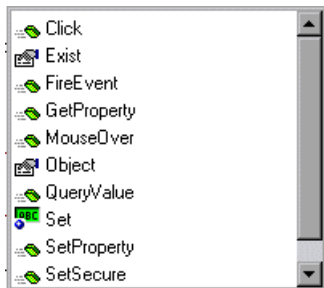
By default, Astra QuickTest displays the syntax for functions as you type. You can disable or enable this **Statement Completion** option in the Editor Options dialog box. For additional information, see Chapter 26, [Customizing the Expert View](#).

To generate a function in the Expert View:

- 1 In the Expert View, type a period after the object upon which you want to perform the function.

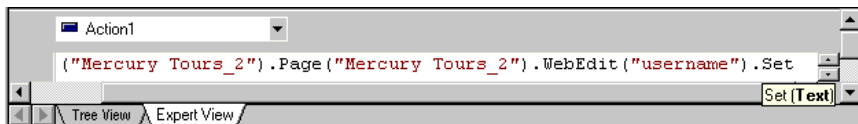


- 2 A list of the available functions for the object is displayed.



Double-click a function in the list. Astra QuickTest inserts the function into the line of script.

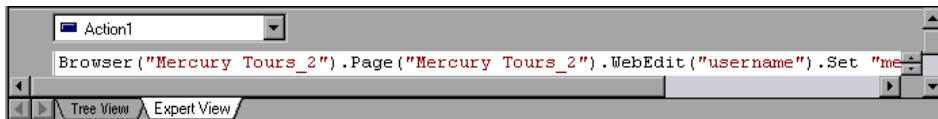
- 3 If the function contains arguments, and the Statement Completion option is enabled, Astra QuickTest displays the syntax of the function.



In the above example, the **Set** function has one argument, called *Text*. The argument name represents the text to enter in the edit box.

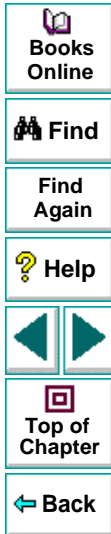


- 4 Insert an argument after the function.



Tip: When programming in VBScript, use parentheses around function parameters if the function is expected to return a value. If the function does not return a value, parentheses are optional.

Note: To find the syntax for a function, refer to the *Astra QuickTest Function Reference*.



Enhancing Tests with Comments, Calculations, and Control-Flow Statements

Astra 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 Astra QuickTest sends to your test results. To improve the readability of your tests, you can also add comments to your test.

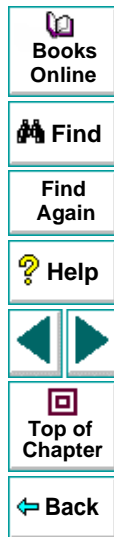
For information on how to use these programming concepts in the Tree View, see Chapter 21, [Enhancing Your Tests with Programming](#).

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, Astra 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 the color green. For example:

```
'Sets the word "mercury" into the "password" edit field.  
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").  
Set "mercury"
```

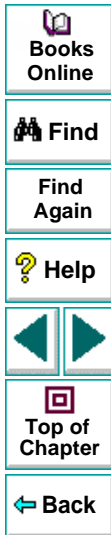
Note: You can also add a comment line using VBScript's **Rem** function. For additional information, refer to the *VBScript Reference*.



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 QueryValue function

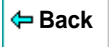
```
passenger = Browser ("Mercury_Tours"). Page ("Find_Flights").  
    WebEdit("numPassengers"). QueryValue("value")
```

'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 Astra 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

<i>counter</i>	The variable used as a counter.
<i>start</i>	The start number of the counter.
<i>end</i>	The last number of the counter.
<i>step</i>	The number to increment at the end of each loop. The default is 1.
<i>statement</i>	The statement to be executed during the loop.

In the following example, Astra QuickTest calculates the factorial value of the number of passengers using the For statement.

```
number = Browser("Mercury Tours").Page("Find  
Flights").WebEdit("numPassengers").QueryValue("value")  
total = 1  
For i=1 to number  
    total = total * i  
next  
MsgBox "!" & number & "=" & total
```



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

For...Each Statement

A **For...Each** loop instructs Astra QuickTest to execute one or more statements for each element in an array. 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 Astra 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

Astra QuickTest calculates the factorial value of the number of passengers using the Do...Loop.

```
number = Browser("Mercury Tours").Page("Find  
Flights").WebEdit("numPassengers").QueryValue("value")  
total = 1  
i = 1  
do while i <= number  
    total = total * i  
    i = i + 1
```

Loop

```
MsgBox "!" & number & "=" & total
```



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

While Statement

A **While** statement instructs Astra 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, Astra QuickTest performs a loop using the **While** statement while the number of passengers is fewer than four. Within each loop, Astra QuickTest increments the number by one.

```
passengers = Browser("Mercury Tours").Page("Find Flights").  
    WebEdit("numpassengers").QueryValue("value")  
while passengers < 4  
    passengers = passengers + 1  
wend
```



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

If...Then...Else Statement

The **If...Then...Else** statement instructs Astra 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, Astra QuickTest closes the browser.

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").QueryValue("value")
if (passengers < 4) then
    Browser("Mercury Tours").close
Else
    Browser("Mercury Tours").Page("Find Flights").Image("continue").Click
69,5
End If
```



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

The 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").QueryValue("value")
```

Note: Astra QuickTest includes Microsoft’s *VBScript Language Reference*. The VBScript Language reference describes VBScript in detail. To open this reference, choose **Help > VBScript Reference**.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Advanced Features

Working with Astra QuickTest—for Power Users

This chapter answers some of the questions that are asked most frequently by *advanced users* of Astra 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**



Recording and Running Tests

- **How does Astra QuickTest capture user processes?**

Astra QuickTest hooks the browser (Netscape or Microsoft Internet Explorer). As the user navigates the Web site, Astra QuickTest intercepts and records all steps as they enter the browser. Astra QuickTest can then run the test by running the steps as they originally occurred.

- **How does Astra QuickTest record and identify objects on Web pages?**

Astra QuickTest can record all Web objects on a Web page. Each HTML tag is considered a Web object. Astra QuickTest identifies each object by its HTML tag and logical name and stores the descriptions of each object in the memory.

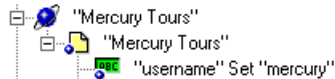
- **How can I record on objects or environments not supported by Astra QuickTest?**

By default, all tests are recorded using the standard recording mode. If you are unable to record on an object in a given environment in the standard mode, or if you want to record mouse clicks with the exact x- and y-coordinates, you may want to record on those objects using *low-level recording*.

While recording, select **Test > Low-Level Recording** in Astra QuickTest. The record mode changes to low-level recording, and all of your clicks are recorded based on mouse coordinates. When Astra QuickTest runs the test, the mouse retraces the recorded movements.

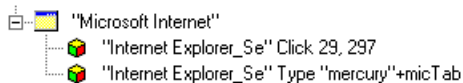


For example, if you type the word `mercury` into a user name edit box and then click the Tab key while in standard recording mode, your test tree and script would appear as follows:



```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set
"mercury"
```

If you perform the same action while in low-level recording mode, Astra 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 Astra QuickTest will recognize them when it runs the test. For more information, see Chapter 4, [Understanding How Astra QuickTest Identifies 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** function to check whether or not a spawned window exists. For example:

```
Browser("Window_logical_name").Exist
```

For additional information about the **Exist** function, refer to the *Astra QuickTest Function Reference*.

- **How does Astra QuickTest record on dynamically generated URLs and Web pages?**

Astra QuickTest actually clicks on links as they appear on the page. Therefore, Astra 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 Astra QuickTest records the “IMG” HTML tag, and the name of the image. This enables Astra QuickTest to find this image in the future and click on it.



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

Advanced Web Issues

- **How does Astra 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.

Astra QuickTest stores cookies in the memory for each user, and the browser handles them as it normally would.

- **How does Astra 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 Astra QuickTest.

- **How does Astra 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 Astra QuickTest or the browser.

- **How does Astra 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, Astra QuickTest has no problem handling meta tags.



- **Does Astra 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 Astra QuickTest.

- **Does Astra QuickTest work with COM?**

Astra QuickTest complies with the COM standard.

Astra QuickTest supports COM objects embedded in Web pages (which are currently accessible only using Microsoft Internet Explorer).

- **Does Astra 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.

Astra QuickTest supports XML and recognizes XML tags as objects. Note that XML is not currently completely supported by Microsoft Internet Explorer and Netscape.



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 Astra 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](#).



Configuring Astra QuickTest



Configuring Astra QuickTest

Setting Astra QuickTest Testing Options

You can control how Astra QuickTest records and runs tests by setting testing options.

This chapter describes:

- **Setting Astra QuickTest Testing Options**
- **Selecting Astra QuickTest Testing Options**



About Setting Astra QuickTest Options

Astra QuickTest testing options affect how you record and run tests. For example, you can set the speed at which Astra QuickTest runs a test, or set the timing-related settings used by Astra QuickTest. The values you set remain in effect for all tests and for subsequent testing sessions.

You can also set testing options for that effect only the test currently open in Astra QuickTest. For more information, see Chapter 25, [Setting Testing Options for a Single Test](#).



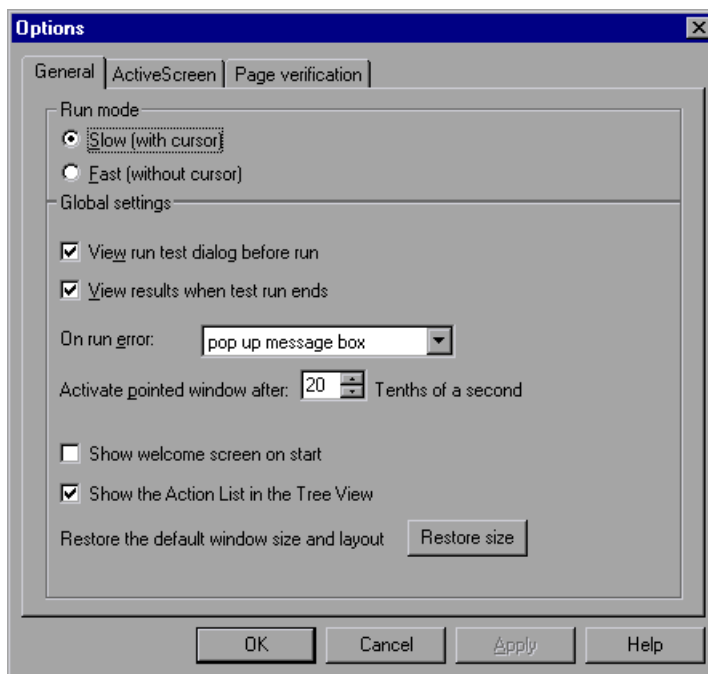
Setting Astra QuickTest Testing Options

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 Astra QuickTest testing options:

- 1 Choose **Tools > Options**.

The Options dialog box opens. It is divided by subject into three tabbed pages.



- 2 To choose a page, click a tab.
- 3 Set an option, as described in [Selecting Astra QuickTest Testing Options](#) on page 462.
- 4 To apply your changes and keep the Options dialog box open, click **Apply**.
- 5 When you are done, click **OK** to apply your changes and close the dialog box.



Selecting Astra QuickTest Testing Options

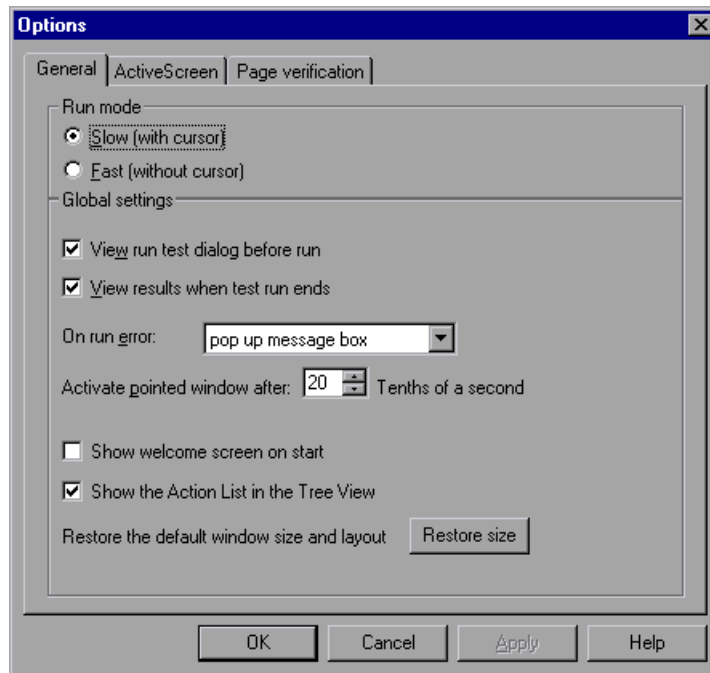
The Options dialog box contains the following tabbed pages:

Tab Heading	Subject
General	options for test run and global settings
Active Screen	options for displaying Web pages in the ActiveScreen
Page Verification	options for page checkpoints



General Testing Options

The General tab options affect how Astra QuickTest runs tests and displays test results.




The General tab includes the following options:

Option	Description
Run mode - Slow (with cursor)	Instructs Astra 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 <i>Astra QuickTest Installation Guide</i> .
Run mode - Fast (without cursor)	Instructs Astra QuickTest to run your test without the execution arrow in the left margin of the test, marking each step or statement as it is interpreted.
View results when test run ends	Instructs Astra QuickTest to display the test results automatically following the test run.
On run error	Determines how Astra 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 run when an error occurs.



Books Online



Find

Find Again



Help





Top of Chapter



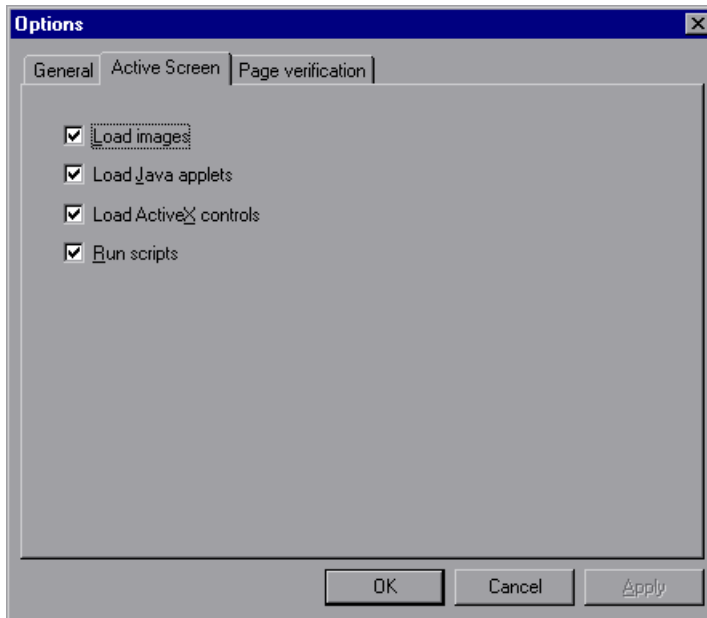
Back

Option	Description
Activate pointed window after	Specifies the time (in tenths of a second) that Astra QuickTest waits before it sets focus on the Web browser.
Show welcome screen on start	Determines whether the Welcome screen is displayed when starting Astra QuickTest.
Show the Action List in the Tree View	Determines whether the Action List is displayed in the Tree View. If your test contains reusable or external actions, this option is automatically enabled.
Restore the default window size and layout	Restores the Astra QuickTest window so that it displays the default panes in the default sizes.



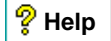
Active Screen Testing Options

The Active Screen tab options affect how Astra QuickTest displays Web pages in the ActiveScreen.



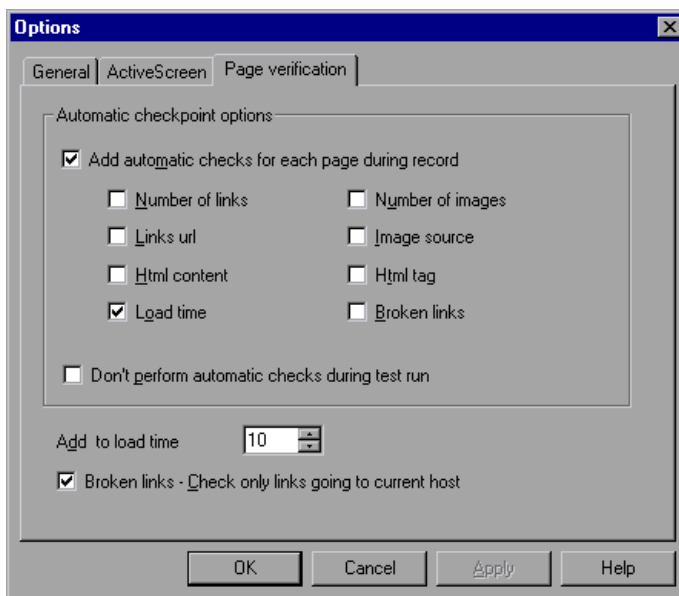
The Active Screen tab includes the following options:

Option	Description
Load images	Instructs Astra QuickTest to load images from your browser page to the ActiveScreen pane.
Load Java applets	Instructs Astra QuickTest to load Java applets from your browser page to the ActiveScreen pane. If this option is cleared, a default Java image appears in the ActiveScreen for all Java applet objects.
Load ActiveX controls	Instructs Astra QuickTest to load ActiveX controls from your browser page to the ActiveScreen pane. If this option is cleared, a default ActiveX image appears in the ActiveScreen for all ActiveX control objects.
Run scripts	Instructs Astra QuickTest to run scripts while loading your browser page on the ActiveScreen pane.



Page Verification Options

The Page Verification tab options affect how Astra QuickTest performs page checkpoints on a Web page.

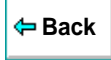


The Page Verification tab includes the following options:

Option	Description
<p>Add automatic checks for each page during record</p>	<p>Instructs Astra QuickTest to add a page checkpoint for each page navigated to in your site. The automatic page checkpoint includes the checks that you select from among the options below it.</p> <p>Note: If you are testing a 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.</p>
<p>Number of links</p>	<p>Checks that the expected number of links is identical to the number that appears in the run session.</p>
<p>Links URL</p>	<p>Checks that the expected URL addresses for the links are identical to the addresses that appear in the run session.</p>
<p>HTML content</p>	<p>Checks that the expected source code is identical to the source that appears in the run session.</p>
<p>Load time</p>	<p>Checks that the amount of time it takes for the page to load during the record session and the run session are identical.</p>
<p>Number of images</p>	<p>Checks that the expected number of images is identical to the number that appears in the run session.</p>



Option	Description
Image source	Checks that the expected source paths of the images are identical to the sources in the run session.
HTML tag	Checks that the expected HTML tags in the source code are identical to those in the run session.
Broken links	Displays the number of broken links that appear during the run session.
Don't perform automatic checks during test run	Instructs Astra QuickTest to ignore the automatically added page checkpoints when running your test.
Add to load time	Instructs Astra QuickTest to add a specified number of seconds to the load time of the page. This option is a safeguard which prevents the test from failing in the event that the amount of time it takes for a page to load during the run session is higher than the amount of time it took during the record session.
Broken links - Check only links going to current host	Instructs Astra QuickTest to check only for broken links that are targeted to your current host.



Configuring Astra QuickTest

Setting Testing Options for a Single Test

You can control how Astra 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**



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 Astra QuickTest to run a parameterized action for only certain lines in the table in the Data pane. You can also teach Astra 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 from within a test, for part of the 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 Astra QuickTest 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 seven 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 474.
- 4 To apply your changes and keep the Test Settings dialog box open, click **Apply**.
- 5 When you are done, click **OK** to apply your changes and close the dialog box.

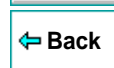
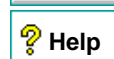


Selecting Testing Options for a Single Test

The Test Settings dialog box contains the following tabbed pages:

Tab Heading	Subject
StartUp	options for selecting the Web browser and initial URL to use as the starting application of the test.
Run	options for setting test iteration preferences
Properties	options for setting the properties of the test
Web	options for setting the way the test records and runs on the Web browser
Object Mapping	options for mapping a custom object to a standard class
Java	options for setting how Astra QuickTest records and runs tests on a Java applet. Note: Testing on Java objects is supported only in Astra QuickTest Professional.
ActiveScreen	options for controlling the behavior of Web pages in the ActiveScreen

This section lists the testing options you can set using the Test Settings dialog box.



StartUp Testing Options

The StartUp tab options set which Web browser to use and whether to use the Web browser that is currently running or to open a new browser window to a specified location when recording and running the test.

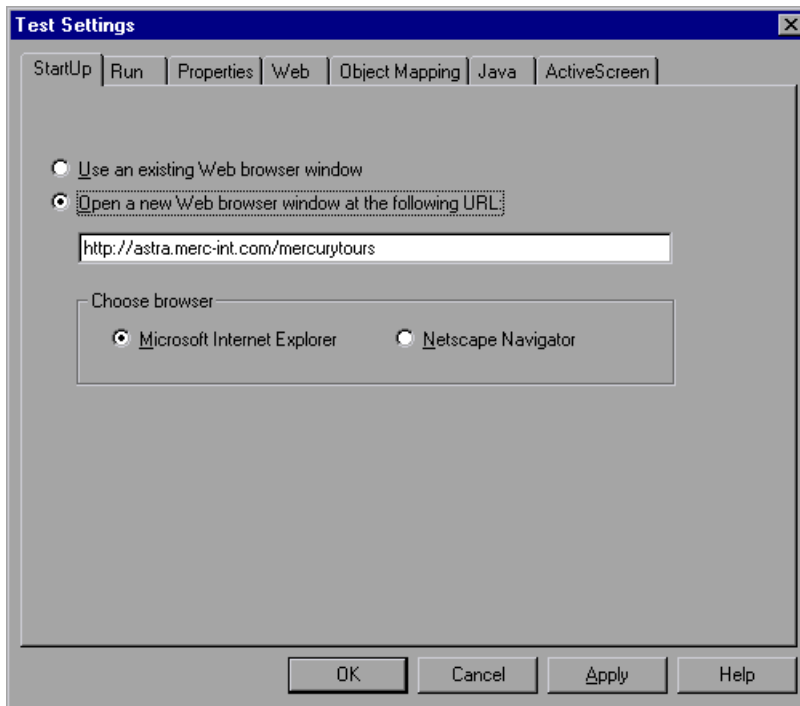
Note: You can also set the Startup options in the Start Recording dialog box, which opens when you start recording.

If you change the startup options for different recording sessions within one test (i.e. when recording a new action), confirm that the Startup testing options are set appropriately for the first step in the test before you run the test.

For more information about the Start Recording dialog box, see [Recording a Test](#) on page 47.



The StartUp tab includes the following options:



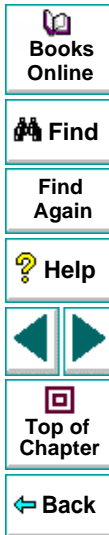
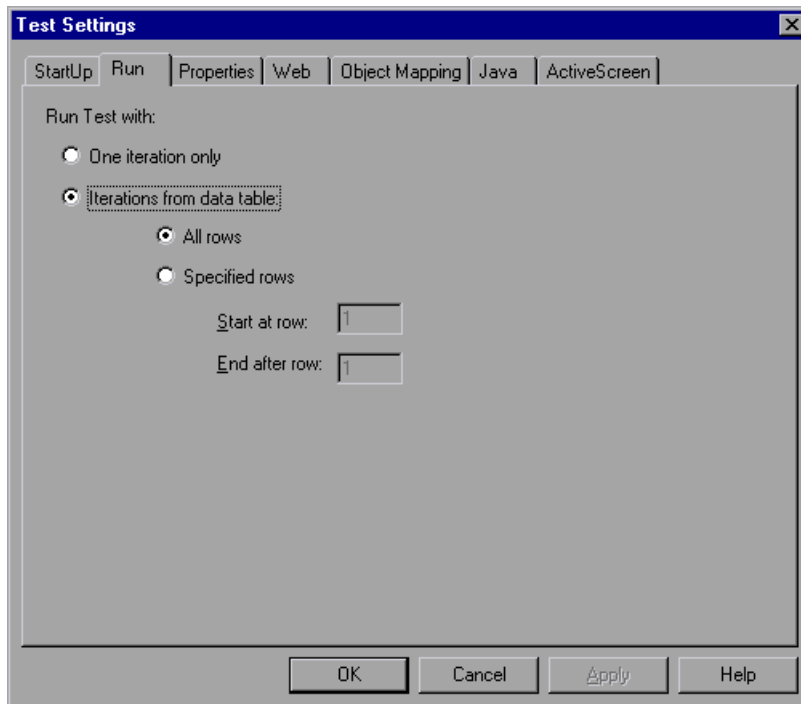
Option	Description
Use existing Web browser window	Instructs Astra QuickTest to use your existing browser window to record and run the test.
Open new Web browser window at the following URL	Instructs Astra QuickTest to open a new browser session to record and run the test using the specified Web location address.
Choose browser	Instructs Astra QuickTest to use the specified browser type to record and run the test.

Run Testing Options

When you run a test, Astra QuickTest performs the steps you recorded on your Web site. When you run a test with global parameters, Astra QuickTest runs the test for each row in the table in the Data pane, using the parameters you specified. If your test includes parameters, you can choose to run a range of data sets. For more information, see Chapter 15, [Working with Data Tables](#).



You can use the Run tab to instruct Astra QuickTest to parameterize a test for only certain lines in the Data pane.



The Run tab includes the following options:

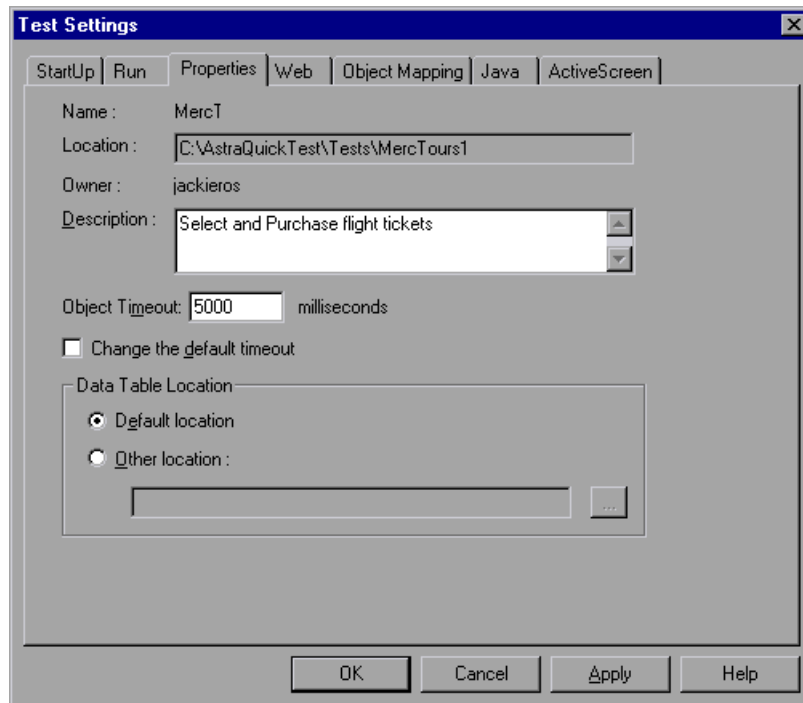
Option	Description
One iteration only	Runs the test only once, using only the first row in the global data table.
Iterations from the data table	Runs the test with iterations on the rows specified in the option(s) below.
All Rows	Runs the test with iterations using all rows in the global data table
Specified Rows	Runs the test with iterations using the values in the global data table for the specified row range. Use Start from row to indicate the row number to start the run. Use Run to row to indicate the row number to end the run.

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 Chapter 14, [Setting the Run Properties for an Action](#).



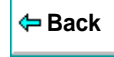
Properties Testing Options

The Properties tab options define general test information.



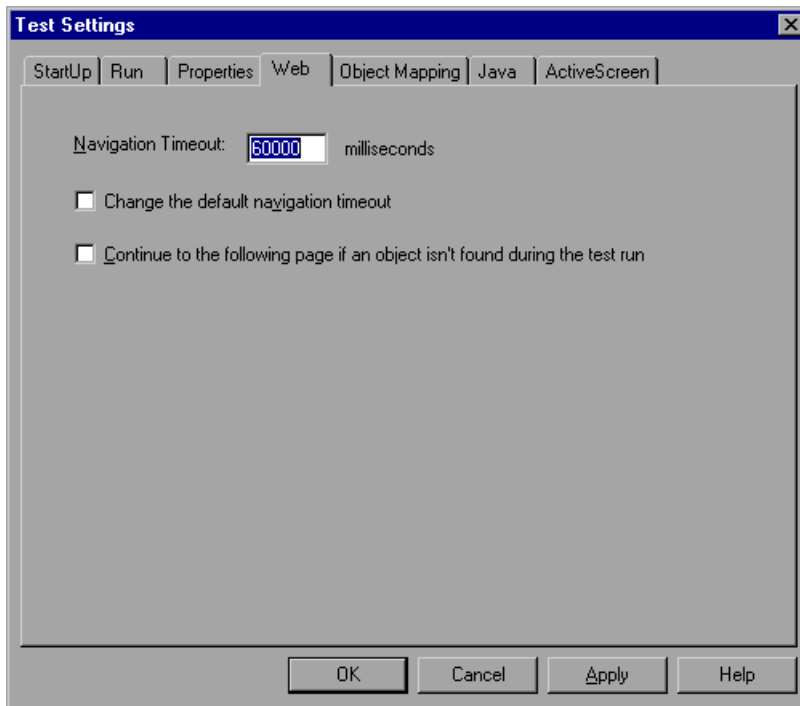
The Properties tab includes the following options:

Option	Description
Name	Indicates the name of the test.
Location	Indicates the path of the test.
Owner	Indicates the user name.
Description	Indicates the test description.
Object timeout	Indicates the maximum interval (in milliseconds) that Astra QuickTest will wait for a Web object to stabilize before running a test step.
Change the default timeout	Changes the default Object timeout for all new tests.
Default Location	Instructs Astra QuickTest to use data stored in the default data table location under the test folder.
Other Location	Instructs Astra QuickTest to use data stored in the specified data table location. The data table can be any Excel (.x/s) file.



Web Testing Options

The Web tab options affect the recording and running of tests on Web objects.



The Web tab includes the following options:

Option	Description
Navigation timeout	Indicates the interval (in seconds) Astra QuickTest waits for the Web page to load before running a test step.
Change the default navigation timeout	Changes the default Navigation timeout globally.
Continue to the following page if an object isn't found during the test run	<p>Instructs Astra 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, thus enabling the test to continue the test run.</p> <p>Note: When Astra 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.</p>

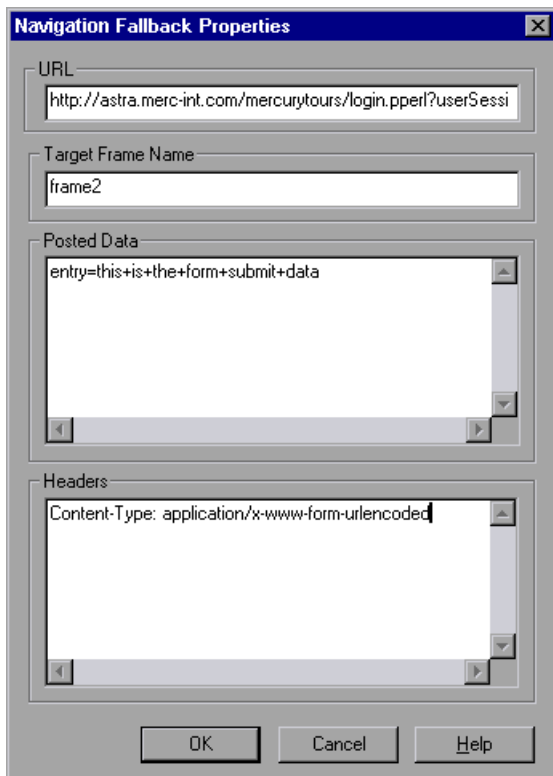
Navigation Fallback Properties

If a navigation step in a page fails during a test run and you have selected the **Continue to the following page if an object isn't found during the test run** option in the Web tab, Astra QuickTest can use the Navigation Fallback Properties as an alternate way to navigate to the next page in the test. You can set the navigation fallback properties for each page in your test.



To set navigation fallback properties:

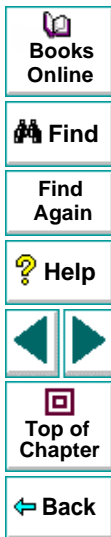
- 1 Right-click on a page or step icon in the test tree view or on a script line in the Expert view.
- 2 Select **Navigation Fallback Properties**. The Navigation Fallback Properties dialog box opens.



The screenshot shows the "Navigation Fallback Properties" dialog box with the following fields:

- URL:** `http://astra.merc-int.com/mercurytrous/login.pperl?userSessi`
- Target Frame Name:** `frame2`
- Posted Data:** `entry=this+is+the+form+submit+data`
- Headers:** `Content-Type: application/x-www-form-urlencoded`

Buttons at the bottom: OK, Cancel, Help.

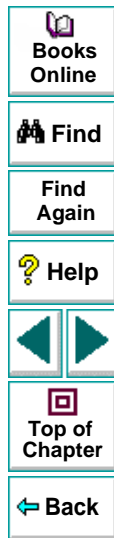


- 3 Enter the page navigation details and click **OK**.
 - **URL** - The complete URL of the next page in the test.
 - **Target Frame Name** - The name of the frame in which the specified URL should be displayed.
 - **Posted Data** - The data to be sent to the server with the HTTP POST transaction. For example, 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`

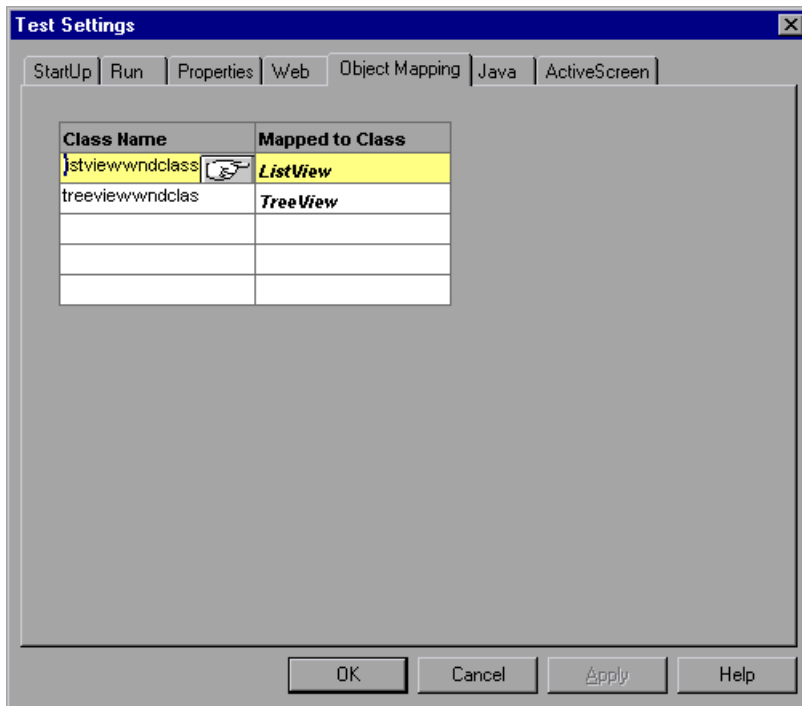
This parameter is ignored if the URL is not an HTTP URL.
- 4 Repeat steps 1 - 3 for each page for which you want to set navigation fallback properties.

Object Mapping Testing Options

The Object Mapping tab options enable you to teach Astra QuickTest a custom object and map it to a standard class. For example, if your site has a custom button that Astra QuickTest cannot identify, a click on this button is recorded as a Web object. You can teach Astra QuickTest the custom class and map it to the *button* class. Then, when you click the button, the operation is recorded as a button click.



Note that a custom object should be mapped only to a standard class with comparable behavior. For example, you cannot map a custom button to the edit class.



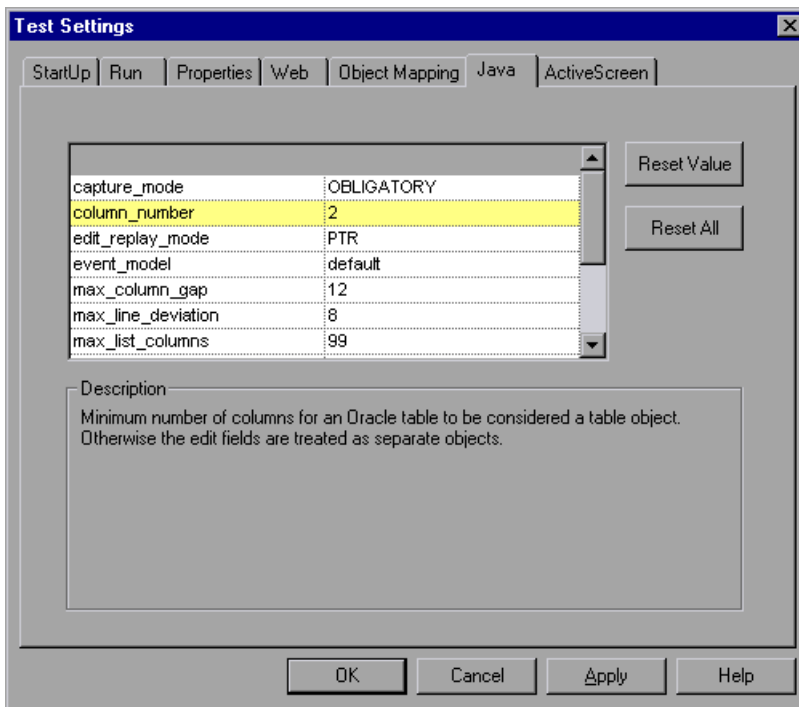
The Object Mapping tab includes the following options:

Option	Description
Class Name	Lists object class names. To add a class name, click a Class Name box and then use the pointing hand to click the object whose class you want to add.
Mapped to Class	Lists standard class names. To map a custom object to a standard class, click a Mapped to Class box to activate the class list. Select the class to which you want to map the custom class.



Java Testing Options

The Java tab option enables you to configure how Astra QuickTest learns the descriptions of Java objects, records tests, and runs tests on Java applets or applications.



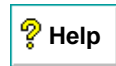
Note: The Java tab is only available when the Java support is installed. Java support is only available in Astra QuickTest Professional.

The Java tab includes the following options:

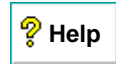
Option	Description
capture_mode	<p>Indicates the quantity level of properties that will be captured for each object in an applet when the applet is captured for the ActiveScreen.</p> <p>“OBLIGATORY” captures fewer properties. Should be used for heavy applets with many objects.</p> <p>“ALL” captures all properties and values of the objects in an applet.</p> <p>Default value: OBLIGATORY</p>
column_number	<p>Indicates the minimum number of columns for a table to be considered a table object. Otherwise the edit fields are treated as separate objects.</p> <p>(Oracle only)</p> <p>Default value: 2</p>



Option	Description
edit_replay_mode	<p>Controls how actions are performed on edit fields. Select one or more of the following values:</p> <p>“S” sends the setValue () method to set a value of the edit object.</p> <p>“P” sends KeyPressed event to the object for every character from the input string.</p> <p>“T” sends KeyTyped events to the object for every character from the input string.</p> <p>“R” sends KeyReleased event to the object for every character from the input string.</p> <p>“F” generates a FocusLost event at the end of function execution.</p>
event_model	<p>Sets the event model that will be used to send events to the objects in the applicatin under test. Choose one of the following values:</p> <p>“NEW” for applications written in the new event model.</p> <p>“OLD” for applications written in the old event model.</p> <p>“DEFAULT” uses the OLD event model for AWT objects and NEW for all other toolkit objects.</p> <p>Default value: "DEFAULT"</p>



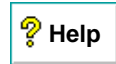
Option	Description
max_column_gap	The maximum number of pixels between objects in a table to be considered a column. (Oracle only) Default value: 12
max_line_deviation	The maximum number of pixels between objects to be considered to be on a single line. (Oracle only) Default value: 8
max_list_columns	The maximum number of columns in an Oracle LOV object to be considered a list. A larger number constitutes a table. (Oracle only) Default value: 99
max_row_gap	The maximum number of pixels between objects to be considered one table row. (Oracle only) Default value: 12
max_text_distance	Sets the maximum distance in pixels, to look for attached text. Default value: 100



Option	Description
record_by_number	<p>Controls how items in list, combo box, and tree view objects are recorded.</p> <p>The variable can be one of the following values: list, combo, tree, or a combination separated by a space.</p> <p>If one of these objects has been detected, numbers are recorded instead of the item names.</p>
table_record_mode	<p>Sets the record mode for a table object (CS or ANALOG).</p> <p>“CS” indicates that the record mode is Context Sensitive.</p> <p>“ANALOG” records only low-level (Analog) table functions: ClickCell, DoubleClickCell, and Drag. (JFC JTable object only).</p> <p>Default value: “CS”</p>

You can reset the value of any Java test setting to its default value by highlighting the value and clicking **Reset Value**.

You can reset the value of all the Java test settings by clicking **Reset All**.

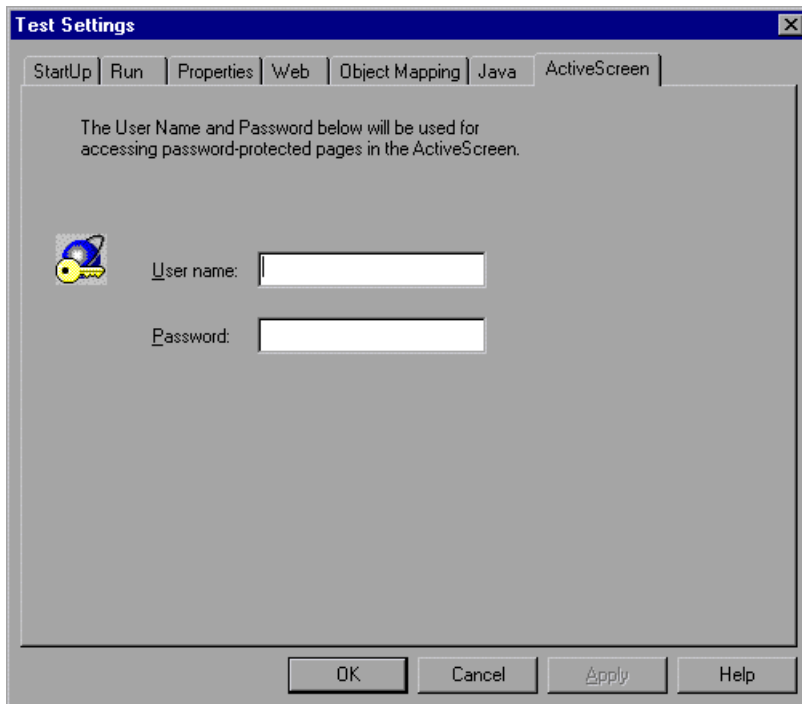


Note: The options in the Java tab can also be set and retrieved from the script using the **SetAUTVar** and **GetAUTVar** functions. For more information see, Chapter 9, [Testing Java Applets](#).



ActiveScreen Testing Options

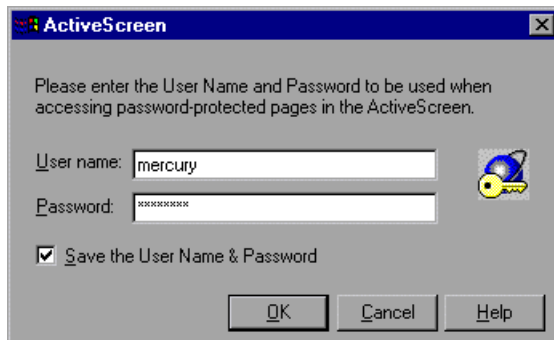
The ActiveScreen tab enables you to enter a User name and password for the test. If an ActiveScreen page requires a User name and password, it will take them from this entry rather than requiring you to enter them each time you want to access that Active Screen page.



The ActiveScreen tab includes the following options:

Option	Description
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 tab 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 user name and password** check box.



Configuring Astra QuickTest

Customizing the Expert View

You can customize the way your test is displayed when you work in the Expert View. Astra 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 appear, and create a list of typing errors that will be automatically corrected by Astra QuickTest.

This chapter describes:

- **Setting Display Options**
- **Personalizing Editing Commands**



About Customizing Your Test in the Expert View

Astra 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 Astra 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.

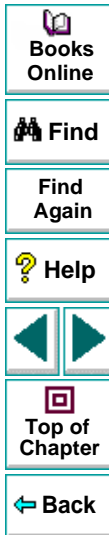
Display options also let you change the color and appearance of different test script elements. These include comments, strings, Astra QuickTest reserved words, operators and numbers. For each test script element, you can assign colors, text attributes (bold, italic, underline), font, and font size. 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 appear when printed.



Personalizing Test Script Editing Commands

Astra 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

Astra QuickTest's display options let you control how test scripts appear in the Expert View tab, how different elements of test scripts are displayed, and how test scripts will appear when they are printed.

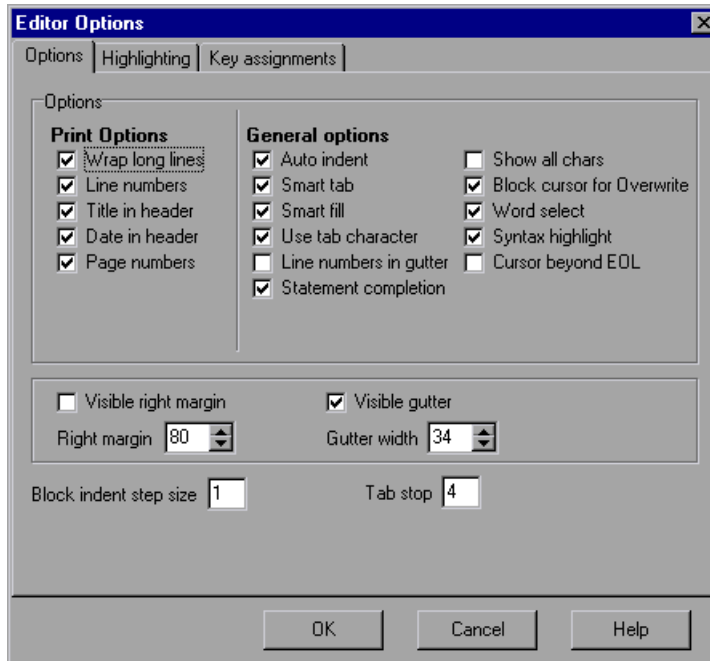
Customizing Test Scripts

You can customize how Astra QuickTest's test scripts are displayed. For example, you can highlight test script elements and show or hide text symbols.

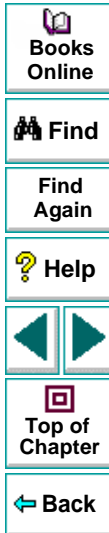


To customize the appearance of your script:

- 1 In the Expert View, choose **Tools > Editor Options**. The Editor Options dialog box opens.



- 2 Click the **Options** tab.



3 Under the **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.
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.



Options	Description
Statement completion	<p>Opens a list box displaying all available matches to the function 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.</p> <p>Displays a tooltip with the function parameters once the complete function name appears in the editor. The function parameters are displayed also whenever the user presses the Ctrl, Shift, and Space keys simultaneously or choose Edit > Parameter Info.</p>
Show all chars	<p>Displays all text symbols, such as tabs and paragraph symbols.</p>
Block cursor for Overwrite	<p>Displays a block cursor instead of the standard cursor when you select overwrite mode.</p>
Word select	<p>Selects the nearest word when you double-click in the Expert View tab.</p>
Syntax highlight	<p>Highlights test script elements such as comments, strings, or reserved words.</p>
Cursor beyond EOL	<p>Enables Astra QuickTest to display the cursor after the end of the current line.</p>
Visible right margin	<p>Displays a line that indicates the Expert View tab's right margin.</p>



Options	Description
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 509.
Tab stop	Sets the distance, in characters, between each tab stop.

Highlighting Script Elements

Astra QuickTest test scripts contain many different elements, such as comments, strings, Astra QuickTest reserved words, operators and numbers. Each element of a Astra 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.



Books
Online



Find

Find
Again



Help



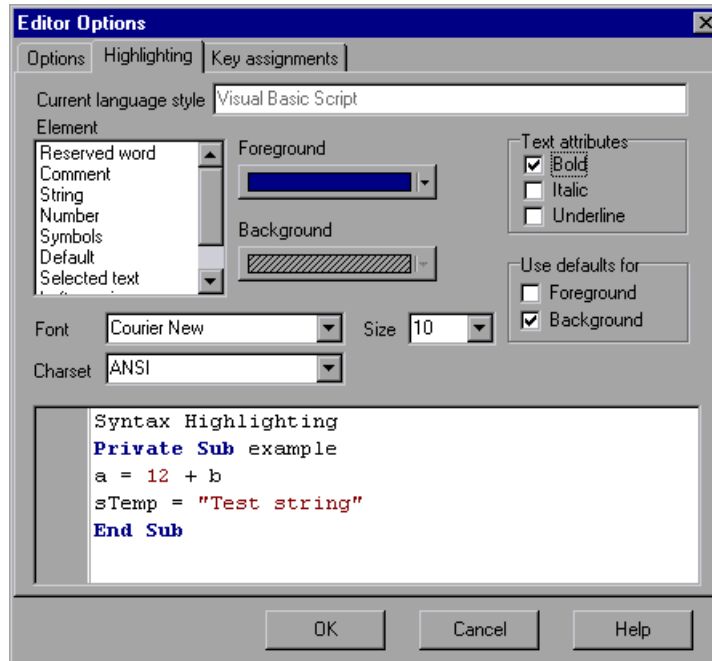
Top of
Chapter



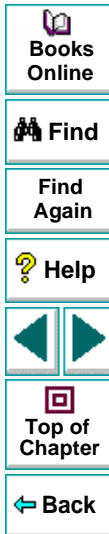
Back

To edit script elements:

- 1 In the Expert View, choose **Tools > Editor Options**. The Editor Options dialog box opens to 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 appears 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.
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

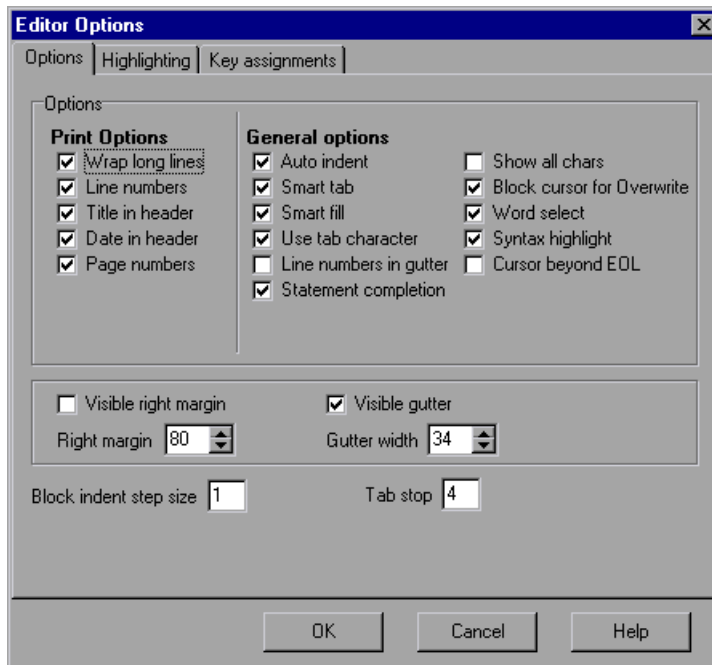
You can set how the hard copy of your script will appear when it is sent to the printer. For example, your printed script can include line numbers, the name of the file, and the date it was printed.

To customize your print options:

- 1 In the Expert View, 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.
File name in header	Inserts the file name 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

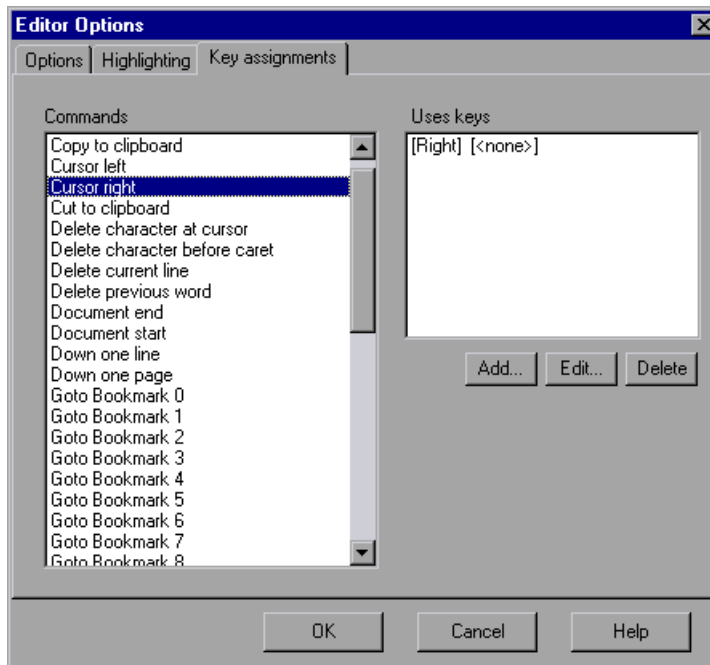
You can personalize the default keyboard commands you use for editing test scripts. Astra QuickTest includes keyboard commands that let you move the cursor, delete characters, cut, copy, and paste information to and from the clipboard. You can replace these commands with your own preferred commands. For example, you could change the Paste command from the default CTRL + V TO CTRL + P.

To personalize editing commands:

- 1 In the Expert View, choose **Tools > Editor Options**. The Editor Options dialog box opens.



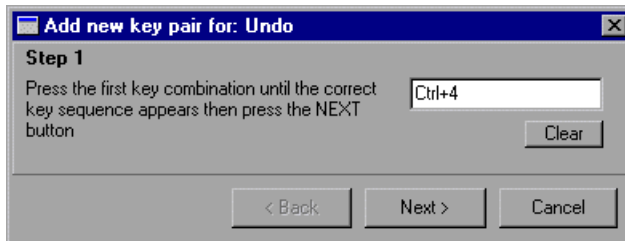
- 2 Click the **Key Assignments** tab.



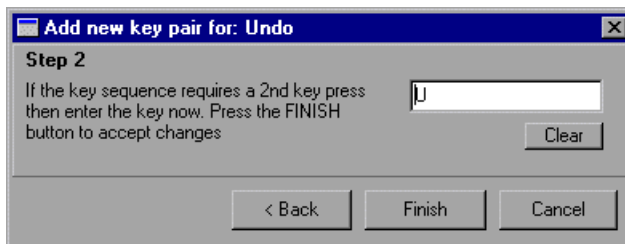
- 3 Select a command from the **Commands** list.



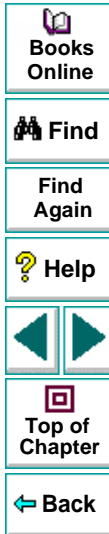
- Click **Add** to create an additional key assignment or click **Edit** to modify the existing assignment. The Add/Edit key pair for dialog box opens. Press the keys you want to use. For example, CTRL + 4.



- Click **Next**. To add an additional key sequence, press the keys you want to use. For example, U.



- 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**.
- Click **OK** to apply the changes.



Configuring Astra QuickTest

Setting Testing Options from a Test Script

You can control how Astra 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 Run-Time Settings**



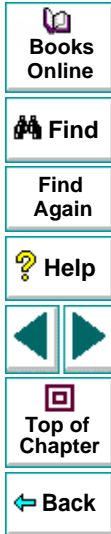
About Setting Testing Options from a Test Script

Astra QuickTest testing options affect how you record test scripts and run tests. For example, you can set the maximum time that Astra 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** function 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**, you can control how Astra 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 Astra QuickTest 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** 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*

The section below provides examples of Astra 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 Don't perform automatic checks during test run option.

AutomaticLinkRun is a global testing option.

Note that you may also set this option using the **Don't perform automatic checks during test run** option in the Page Verification tab of the Options dialog box as described in [Page Verification Options](#) on page 468.

DefaultLoadTime

Sets or retrieves the setting for the add to load time option (in seconds).

DefaultLoadTime is a global testing option.



Note that you may also set this option using the **Add to load time** option in the Page Verification tab of the Options dialog box as described in [Page Verification Options](#) on page 468.

DefaultTimeOut

Sets or retrieves the delay (in milliseconds) for finding objects .

DefaultTimeOut is a per-test setting.

Note that you may also set this option using the **Object Timeout** option in the Properties tab of the Test Settings dialog box as described in [Web Testing Options](#) on page 482.

WebTimeOut

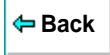
Sets or retrieves the delay (in milliseconds) for navigating to a URL address.

WebTimeOut is a per-test setting.

Note that you may also set this option using the **Navigation Timeout** option in the Web tab of the Test Settings dialog box as described in [Web Testing Options](#) on page 482.

For example, if you execute the following statement:

```
Setting("AutomaticLinkRun")=0
```



Astra 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 **Don't perform automatic checks during test run** check box in the Page Verification tab of the Options dialog box (**Tools > Options**).

Using the **Setting** object with a global testing option changes a testing option globally, and this change is reflected in the Options dialog box. Using the **Setting** object to set test-specific 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 518.



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:

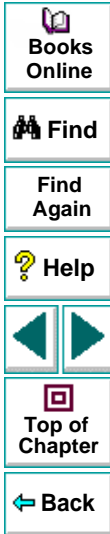
MsgBox (**Setting** (*testing_option*))

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 on [page 514](#).



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 Run-Time Settings

In addition to the global and test-specific settings, you can also add, modify, and remove your own run-time settings. These settings are applicable during the test run only.

To add a new run-time setting, use the syntax:

Setting.Add (*testing_option*, *value*)

For example, you could create a setting that indicates the name of the current tester and writes the name in the report.

```
Setting.Add ("Tester Name", "Mark Train")
Reporter.ReportEvent 1, "Test Run By:", paramcount
```

To modify a run-time setting that has already been initialized, use the same syntax you use for setting any standard setting option:

Setting (*testing_option*) = *new_value*

For example:

```
Setting("Tester Name")=Alice Wonderlin
```

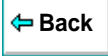


To remove a run-time setting, use the following syntax:

Setting.Remove (*testing_option*)

For example:

Setting.Remove ("Tester Name")



Working with TestDirector



Working with TestDirector

Managing the Testing Process

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 Astra QuickTest with TestDirector**
- **Connecting to and Disconnecting from a Project**
- **Saving Tests to a Project**
- **Opening Tests in a Project**
- **Running Tests from TestDirector**



About Managing the Testing Process

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 cover 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 modes of operation including Requirements manager, Test Plan manager, Test Lab manager, and Defects manager.

Modes of operation

Name	Status	Execution Date	Designer
Verify Defaults	N/A	11/30/1999	mike
Change From Location	N/A	11/30/1999	carol
Insert New Order	N/A	11/30/1999	shelly
Customer Boundary	N/A	11/30/1999	alec
Date Boundary	N/A	11/30/1999	alec
Open Unique Order	N/A	11/30/1999	admin
OrderNo Boundary	N/A	11/30/1999	mike
Check Cancel	N/A	11/30/1999	alec
Verify Class	N/A	11/30/1999	admin
Verify Tickets No.	N/A	11/30/1999	admin

- Books Online
- Find
- Find Again
- Help
-
- Top of Chapter
- Back

The following table describes how you can use each operation mode:

Operation Mode	Description
Requirements manager	Specify testing requirements. This includes defining what you are testing, defining requirement topics and items, and analyzing the requirements.
Test Plan manager	Develop a test plan. This includes defining goals and strategy, dividing your plan into categories, developing tests, recording tests in Astra QuickTest, and analyzing the plan.
Test Lab manager	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 Astra QuickTest, and analyzing test results.
Defects manager	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*.

Note: The integration of Astra QuickTest with TestDirector (described in this chapter) is valid only for TestDirector 7.0i.



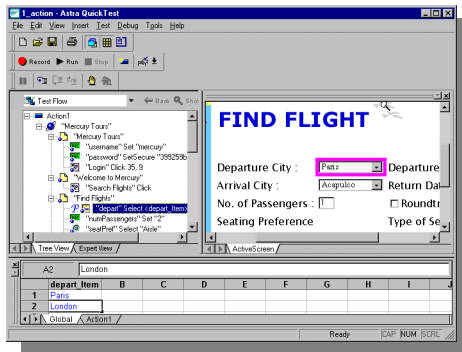
Using Astra QuickTest with TestDirector

Before you start the testing process, you need to create a *TestDirector project*. A *TestDirector* 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.

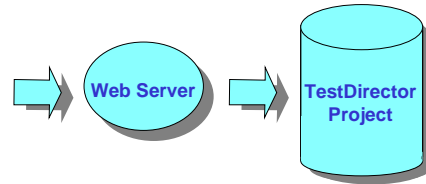
TestDirector and Astra QuickTest work together to integrate all aspects of the testing process. In Astra QuickTest, you can create tests and save them in your TestDirector project. After you run your tests, you can view the results in TestDirector.






In order for Astra QuickTest to access the project, you must connect it to the Web server where TestDirector is installed.



Astra QuickTest



When Astra QuickTest is connected to TestDirector, you can save a test by associating it with a subject in the Test Plan manager. You can schedule to run a test on local or remote hosts. Test run results are sent directly to your TestDirector project.

-  Books Online
-  Find
-  Find Again
-  Help
- 
-  Top of Chapter
-  Back

Connecting to and Disconnecting from a Project

If you are working with both Astra QuickTest and TestDirector, Astra QuickTest can communicate with your TestDirector project. You can connect or disconnect Astra QuickTest from a TestDirector project at any time during the testing process. However, do not disconnect Astra QuickTest from TestDirector while an Astra QuickTest test is opened from TestDirector.

The connection process has two stages. First, you connect Astra QuickTest to the TestDirector server. This server handles the connections between Astra QuickTest and the TestDirector project.

Next, you choose the project you want Astra QuickTest to access. The project stores tests and test run information for the Web site you are testing. Note that TestDirector project databases are password protected, so you must provide a user name and a password.

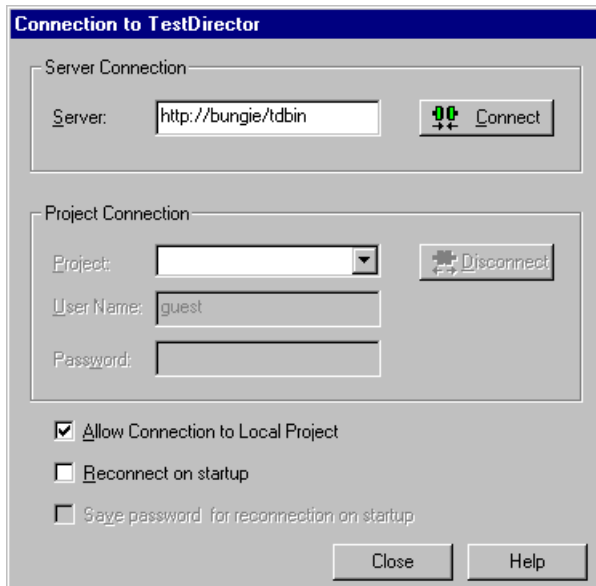
Connecting Astra QuickTest to TestDirector

You must connect Astra QuickTest to the Web server where TestDirector is installed, before you connect Astra QuickTest to a project. For more information, see [Using Astra QuickTest with TestDirector](#) on page 527.



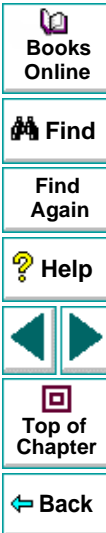
To connect Astra QuickTest to TestDirector:

- 1 Choose **Tools > TestDirector Connection**. The Connection to TestDirector 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: The Web server must be accessible via the Local Area Network (LAN).



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 Type a user name in the **User Name** box.

6 Type a password in the **Password** box.

7 Click **Connect** to connect Astra 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 Start Up** check box.

If the **Reconnect on Start Up** check box is selected, then the **Save Password for Reconnection on Start Up** check box is enabled. To save your password for reconnection on startup, select the **Save Password for Reconnection on Start Up** check box. If you do not save your password, you will be prompted to enter it when Astra QuickTest connects to TestDirector on startup.

8 Click **Close** to close the Connection to TestDirector dialog box.

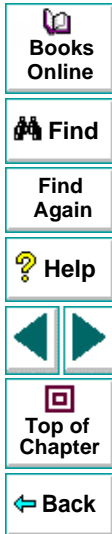
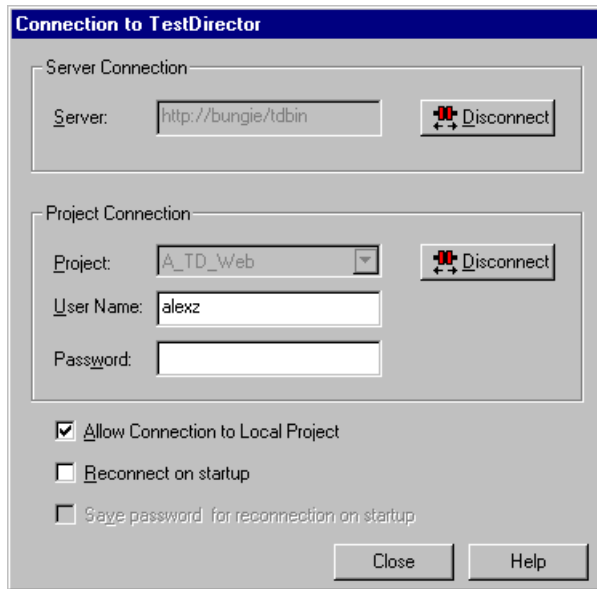


Disconnecting from a TestDirector Project

You can disconnect from a TestDirector project. This enables you to select a different project while using the same server connection.

To disconnect Astra QuickTest from a TestDirector project:

- 1 Choose **Tools > TestDirector Connection**. The Connection to TestDirector dialog box opens.



- 2 In the **Project Connection** section, click **Disconnect** to disconnect Astra QuickTest from the selected project.
- 3 Click **Close** to close the Connection to TestDirector 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 Astra QuickTest from a TestDirector server:

- 1 Choose **Tools > TestDirector Connection**.
The Connection to TestDirector dialog box opens.
- 2 In the **Server Connection** section, click **Disconnect** to disconnect Astra QuickTest from the TestDirector server.
- 3 Click **Close** to close the Connection to TestDirector dialog box.

Note: If you disconnect Astra QuickTest from a TestDirector server without first disconnecting from a project, Astra QuickTest's connection to that database is automatically disconnected.



Saving Tests to a Project

When Astra QuickTest is connected to a TestDirector project, you can create new tests in Astra 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.

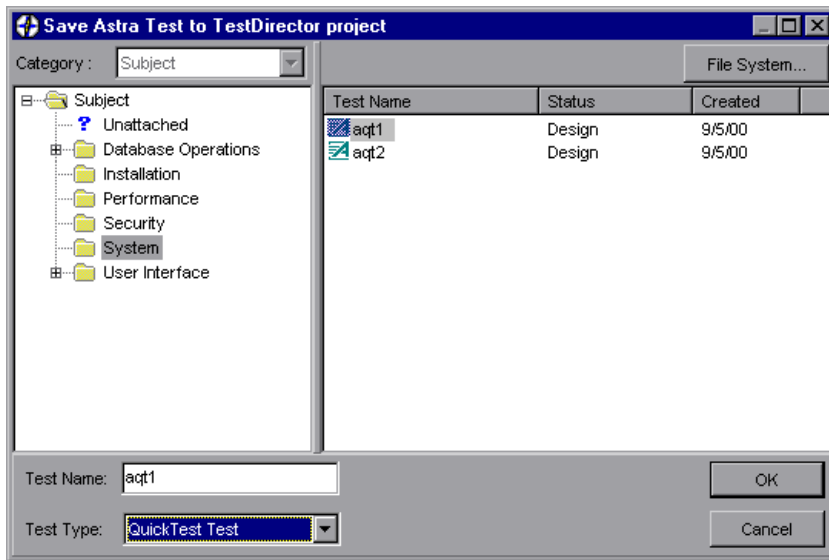
To save a test to a TestDirector project:



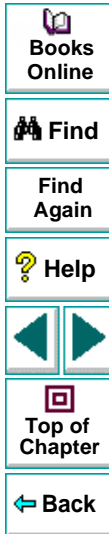
- 1 In Astra QuickTest, click **Save** or choose **File > Save** to save the test.



The Save Astra Test to TestDirector Project dialog box opens and displays the test plan tree.



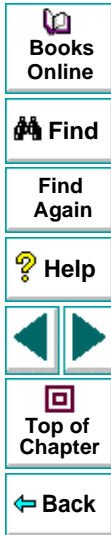
Note that the Save Astra Test to TestDirector Project dialog box opens only when Astra QuickTest is connected to a TestDirector project.



To save a test directly in the file system, click the **File System** button to open the Save Astra QuickTest Test dialog box. (From the Save Astra Test dialog box, you may return to the Save Astra 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** text 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 to close the dialog box.

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

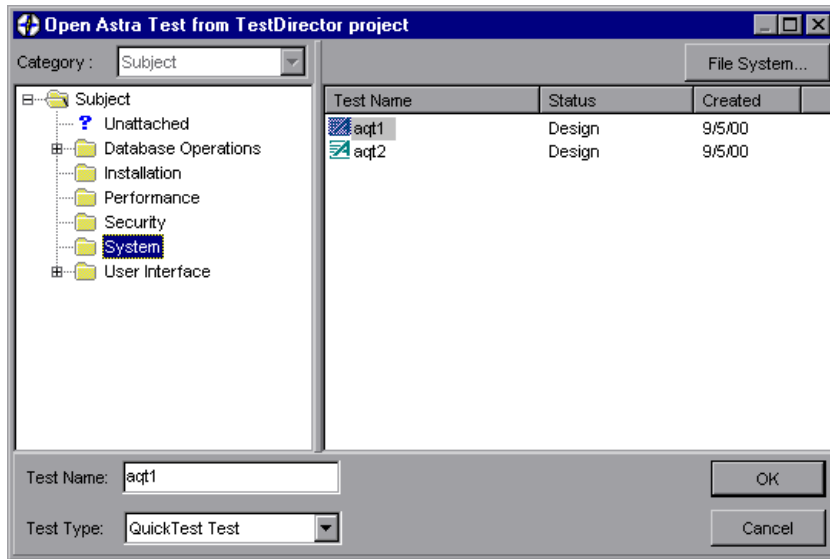
If Astra QuickTest is connected to a TestDirector project, you can open automated tests that are a part of your database. You locate tests according to their position in the test plan tree, rather than by their actual location in the file system.



To open a test saved to a TestDirector project:



- 1 In Astra QuickTest, click **Open** or choose **File > Open** to open the test. The **Open Astra Test** dialog box opens. The Open Astra Test from TestDirector Project dialog box opens and displays the test plan tree.



Note that the Open Astra Test from TestDirector Project dialog box opens only when Astra QuickTest is connected to a TestDirector project.



Books
Online



Find

Find
Again



Help



Top of
Chapter



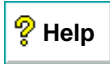
Back

To open a test directly from the file system, click the **File System** button to open the Open Astra QuickTest Test dialog box. (From the Open Astra Test dialog box, you may return to the Open Astra 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. The test opens in a window in Astra QuickTest.



Running Tests from TestDirector

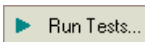
You can run an Astra QuickTest test from a TestDirector project.

Note: If your Astra QuickTest application is installed on Windows 95/98, you must run the Remote Agent on the Astra QuickTest machine before running the test from TestDirector.

To run the Remote Agent, choose **Programs > Astra 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:

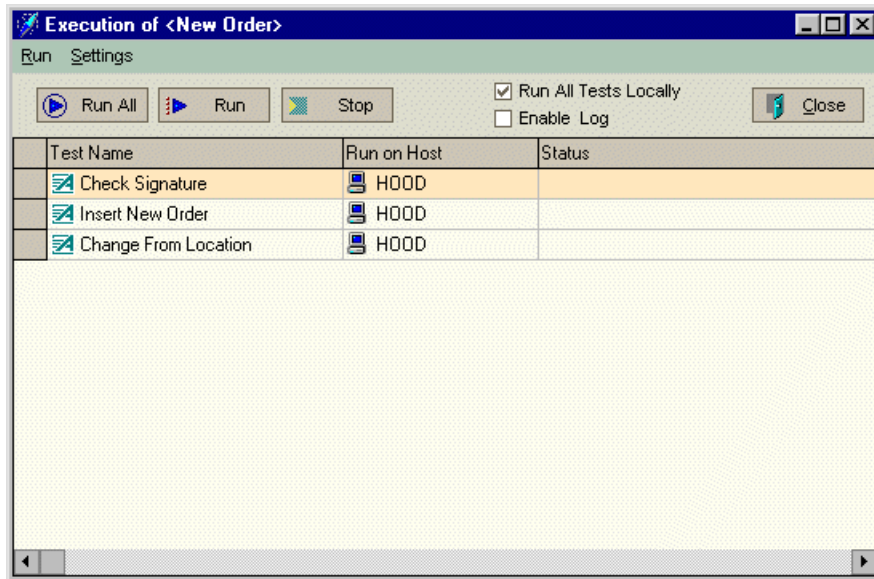
- 1 In TestDirector, click the **Test Lab** manager.
- 2 You can run all the tests in the test set or select specific tests:



- 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, you can view a summary of test results in the Execution Grid tab in TestDirector by clicking the **Launch Report** button on the Test Lab tab. If the Launch Report button is not visible, select **View > Show Last Run Results**.

Launch Report

- 6 Click **Close** to close the Execution dialog box.



Books
Online



Find

Find
Again



Help



Top of
Chapter

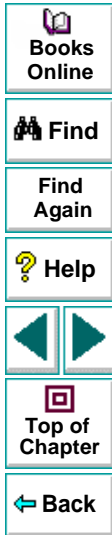


Back

Index

A

- action data sheet **308**
- Action List, definition **32**
- Action tab, Data pane **35, 272**
- action view, definition **275**
- actions **268–303**
 - creating new **287**
 - diagram **274, 290**
 - guidelines for working with **302**
 - multiple actions in a test **270–273**
 - overview **269**
- Activate Pointed Window After box **465**
- ActiveScreen
 - changing **62**
- ActiveScreen tab
 - in Options dialog box **466**
 - in Test Settings dialog box **494**
- Add Automatic Checks for Each Page During
 - Record check box **469**
- Add to Load Time box **470**
- Add/Remove Properties dialog box **73, 200, 250**
- adding checkpoints
 - objects **108–114**
 - pages **83–97**
 - tables **115–120**
 - text **98–107**
- analyzing test results **358–371**
 - checkpoints **367**
 - filtering results **365**
 - printing results **371**
 - Runtime Data table **369**
 - Test Results button **363**
 - Test Results window **360**
- Applets *see* Java applets
- application, sample **14, 24**
- ASCII **309**
- Astra QuickTest
 - Action List **29**
 - Data pane **29**
 - Debug toolbar **29, 38**
 - Display pane **29**
 - File toolbar **29, 37**
 - introduction **18–25**
 - Main toolbar **29, 38**
 - menu bar **29**
 - overview **26–42**
 - resources **14**
 - starting **27**
 - status bar **29**
 - Test pane **29**
 - testing process **20**
 - title bar **29**
 - Welcome window **27**
 - window **29**



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Astra QuickTest help [14](#)
 Astra QuickTest Readme file [14](#)
 Astra QuickTest Test Batch Runner [355–357](#)
 Astra QuickTest testing options [458–470](#)
 Astra QuickTest testing process [20](#)
 Astra QuickTest Tutorial [14](#)
 AutoFill List command, data table [315](#)
 automatic page checkpoints [83](#)
 AutomaticLinkCheck testing option [514](#)
 AWT [154](#)

B

Basic event recording configuration level [387](#)
 behavior, definition [397](#)
 books online [14](#)
 breakpoints
 Clear All Breakpoints button [377](#)
 deleting [377](#)
 Insert/Remove Breakpoint button [377](#)
 overview [373](#)
 setting [376](#)
 Toggle Breakpoint button [376](#)
 Broken Links - Check only links going to current
 host check box [470](#)
 Broken Links check box [470](#)
 bubbling, definition [399](#)
 buttons, Astra QuickTest toolbars [37](#)

C

calculations, in the Expert View [442](#)
 Case Insensitive Ignore Spaces verification
 databases [134](#)
 Case Insensitive verification
 databases [134](#)
 Case Sensitive Ignore Spaces verification
 databases [134](#)
 Case Sensitive verification
 databases [134](#)
 Change the Default Navigation Timeout check
 box [483](#)
 Change the Default Timeout check box [481](#)
 checking databases [121–137](#)
 overview [122–123](#)
 See also databases *and* database
 checkpoints
 checking Web objects [81–120](#)
 Checkpoint command [41](#)
 Checkpoint Properties dialog box [77, 111](#)
 checkpoints [81–120](#)
 checking objects [108–114](#)
 checking pages [83–97](#)
 checking tables [115–120](#)
 checking text [98–107](#)
 Checkpoint Properties dialog box [111](#)
 database [121–137](#)
 definition [51, 75](#)



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- checkpoints (*cont'd*)
 - in the Expert View [425](#)
 - Macromedia Flash objects [176](#)
 - modifying [80](#)
 - Page Checkpoint Properties dialog box [84, 85](#)
 - parameterizing [209–211](#)
 - Table Checkpoint Properties dialog box [117](#)
 - Text Checkpoint Properties dialog box [100](#)
 - Use Data Table Formula option [325](#)
- checks on databases [124–129](#)
- Choose browser option [477](#)
- Class Name option [487](#)
- Clear All Breakpoints button [38, 377](#)
- Clear All Breakpoints command [42](#)
- Clear command, data table [313](#)
- ClickCell function [162](#)
- Collapse All command [364](#)
- column_number Java testing option [489](#)
- Command tab, Debugger pane [36](#)
- Command tab, Debugger view [380](#)
- commands using shortcut keys [39–42](#)
- comments
 - in the Expert View [441](#)
 - in the Tree View [419](#)
- Complete Word command [40](#)
- conditional statements [412](#)
- configuration levels
 - custom [389–400](#)
 - standard [387–388](#)
- configuring event recording [385–402](#)
- connecting Astra QuickTest to a TestDirector project [529–533](#)
- connection string, specifying for database checkpoints [127](#)
- Connection to TestDirector dialog box [530](#)
- Content property check on databases [124–129](#)
- Continue to the following Page if an Object isn't Found During the Test Run check box [483](#)
- conventions. *See* typographical conventions
- Copy command [40](#)
- Copy command, data table [313](#)
- Create Database Checkpoint dialog box [128, 130](#)
- creating a new test [63](#)
- creating checkpoints [74–80](#)
- creating tests [44–64](#)

Books
Online

Find

Find
Again

Help

Top of
Chapter

← Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- Custom event recording configuration
 - adding listening events [395](#)
 - adding objects to the custom list [393](#)
 - deleting objects from the custom list [394](#)
 - setting [390](#)
 - specifying listening criteria [397](#)
- custom event recording configuration [389–400](#)
- Custom Web Event Recording Configuration dialog box [390](#)
- customizing test scripts [496–511](#)
 - highlighting script elements [503](#)
 - overview [497](#)
 - print options [506](#)
 - script window customization [508](#)
- Cut command [40](#)
- Cut command, data table [313](#)

D

- Data menu commands, Data table [315](#)
- Data pane [29, 34](#)
 - Action tab [35, 272](#)
 - Global tab [34, 272](#)
 - table columns [192](#)
 - table rows [193](#)

- data sheets
 - global [307](#)
 - local [308](#)
- data table [304–334](#)
 - AutoFill List command [315](#)
 - Clear command [313](#)
 - Copy command [313](#)
 - Currency(0) command [316](#)
 - Currency(2) command [316](#)
 - Custom Number command [316](#)
 - Cut command [313](#)
 - Data menu commands [315](#)
 - data sheets [307](#)
 - Date (M/d/yy) command [316](#)
 - Delete command [313](#)
 - Edit menu commands [313](#)
 - editing tables [309–316](#)
 - Export command [312](#)
 - File menu commands [312](#)
 - Fill Down command [314](#)
 - Fill Right command [313](#)
 - Find command [314](#)
 - Fixed command [316](#)
 - Format menu commands [316](#)
 - Fraction command [316](#)
 - General command [316](#)
 - Go To command [314](#)
 - Import command [312](#)



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- data table (*cont'd*)
 - Import from command **315**
 - importing data in ASCII **309**
 - importing data in Microsoft Excel 95 **309**
 - importing data in Microsoft Excel 97 **309**
 - Insert command **313**
 - local data sheets **308**
 - menu commands to edit tables **311**
 - Paste command **313**
 - Paste Values command **313**
 - Percent command **316**
 - Print command **312**
 - Recalc command **315**
 - Replace command **314**
 - Scientific command **316**
 - scripting functions, using **327–334**
 - Sort command **315**
 - Time (hmm AM/PM) command **316**
 - using formulas in **323–326**
 - Validation Rule command **316**
 - See also* Data pane
- Data Views button **37**
- database check
 - manually defining an SQL statement **124–129**
 - with Microsoft Query **124–129**
- Database Checkpoint command **125**
- Database Checkpoint wizard **125**
- database checkpoints
 - modifying **137**
- databases
 - Case Insensitive Ignore Spaces verification **134**
 - Case Insensitive verification **134**
 - Case Sensitive Ignore Spaces verification **134**
 - Case Sensitive verification **134**
 - checking **121–137**
 - connection string **127**
 - creating a query in ODBC/Microsoft Query **322**
 - creating a query with Microsoft Query/ SQL statement **128–129**
 - creating checkpoints on **124–129**
 - editing the expected data **132**
 - expected data **122**
 - modifying checkpoints **137**
 - Numeric Content verification **135**
 - Numeric Range verification **135**
 - overview **122–123**
 - result set **124**
 - Specify SQL statement screen **127**
 - specifying which cells to check **131**
 - verification type **134**



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- DataTable object
 - default property [328](#)
 - DeleteSheet function [329](#)
 - Export function [330](#)
 - functions and properties of [328](#)
 - GetCurrentRowt function [330](#)
 - GetRowCount function [329](#)
 - GetSheet function [329](#)
 - GetSheetCount function [329](#)
 - GlobalSheet property [329](#)
 - Import function [330](#)
 - LocalSheet property [329](#)
 - RawValue property [329](#)
 - SetCurrentRow function [330](#)
 - SetNextRow function [330](#)
 - SetPrevRow function [330](#)
 - Value property [328](#)
- Debug toolbar, Astra QuickTest window [29](#), [38](#)
- Debugger Pane [36](#)
- Debugger pane
 - Command tab [36](#)
 - Variables tab [36](#)
 - Watch Expressions tab [36](#)
- debugging tests [372–383](#)
 - Debugger view [377–380](#)
 - Command tab [380](#)
 - Variables tab [379](#)
 - Watch Expressions tab [379](#)
 - deleting breakpoints [377](#)
 - example [381](#)
 - overview [373](#)
 - pausing runs [375](#)
 - setting breakpoints [376](#)
 - Step Into button [374](#)
 - Step Out button [374](#)
 - Step Over button [374](#)
- Default Location option [481](#)
- default properties, modifying [65–73](#)
- DefaultLoadTime testing option [514](#)
- DefaultTimeOut testing option [515](#)
- Delete command [40](#)
- Delete command, data table [313](#)
- deleting breakpoints [377](#)
- Description option [481](#)
- descriptions, modifying [71–73](#)
- Dim statement, in the Expert View [449](#)
- disconnecting from a TestDirector
 - project [532](#)
 - server [533](#)
- Display pane [29](#)
- Display Views button [37](#)

Books
Online

Find

Find
Again

Help

Top of
Chapter

Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Do Not Run Iterations option [285, 479](#)
 Do...Loop statement, in the Expert View [446](#)
 Don't Perform Automatic Checks during Test
 Run check box [470](#)
 dynamic descriptions of objects [70](#)

E

Edit Database Checkpoint dialog box [130](#)
 editing the expected data [132](#)
 for checking databases [130–133](#)
 specifying which cells to check [131](#)
 verification method [135](#)
 verification type [134](#)
 Edit menu commands, data table [313](#)
 edit_replay_mode Java testing option [490](#)
 encoding passwords [207](#)
 Enter network password dialog box [495](#)
 event configuration [385–402](#)
 event_model Java testing option [490](#)
 Excel formulas [323, 325](#)
 Exception Editor [336](#)
 exception handling [335–344](#)
 adding new exceptions [342](#)
 changing status of exceptions [338](#)
 deleting exceptions [344](#)
 Exception Editor [336](#)
 modifying exceptions [340](#)

Expand All command [364](#)
 expected data for database checkpoints [122](#)
 expected value, modifying [79](#)
 Expert View [33, 420–449](#)
 checkpoints [425](#)
 functions [436](#)
 objects and functions [429](#)
 parameters [426](#)
 viewing steps [422](#)
 Export command, data table [312](#)

F

File menu commands, data table [312](#)
 File toolbar, Astra QuickTest window [29, 37](#)
 Fill Down command, data table [314](#)
 Fill Right command, data table [313](#)
 Filter button [365](#)
 Filter Image Check dialog box [95](#)
 Filter Link Checks dialog box [92](#)
 filtering
 hypertext links to check [92](#)
 image sources to check [95](#)
 Find command [40](#)
 Find command, data table [314](#)
 Flash objects *see* Macromedia Flash objects
 For...Each statement, in the Expert View [445](#)
 For...Next statement, in the Expert View [444](#)



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Format menu commands, data table **316**
 formulas
 using in checkpoints **325**
 using in the data table **323–326**
 using to create input parameters **323**
 Function Arguments command **39**
 Function Arguments dialog box **204**
 Function Reference, Astra **14**
 Function wizard **404, 405–411**
 functions
 Expert View **429, 436**
 in the Tree View **405–411**

G

General tab, Options dialog box **463**
 global data sheet **307**
 global parameter, defined **197**
 Global tab, Data pane **34, 272**
 Go To command **40**
 Go To command, data table **314**
 GUI Spy **66**

H

handler, definition **397**
 handling exceptions **335–344**
 adding new exceptions **342**
 changing status of exceptions **338**
 deleting exceptions **344**
 Exception Editor **336**
 modifying exceptions **340**
 help **14**
 High event recording configuration level **387**
 HTML Content check box **469**
 HTML Tag check box **470**
 hypertext links, filtering **92**

I

identifying objects **65–73**
 If Statement dialog box **415**
 If...Then...Else statement, in the Expert View
 448
 Image Source check box **470**
 image sources, filtering **95**
 Import command, data table **312**
 Import from command, data table **315**
 Insert > Database Checkpoint command **125**
 Insert Checkpoint button **38**
 Insert command, data table **313**
 Insert/Remove Breakpoint button **377**



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

J

Java

- Activate function [161](#)
- ActivateCell function [161](#)
- ActivateColumn function [161](#)
- ActivateRow function [161](#)
- AddRange function [162](#)
- AddSelection function [162](#)
- Click function [162](#)
- Close function [162](#)
- Collapse function [162](#)
- DbClick function [163](#)
- Delete function [163](#)
- DeselectCell function [163](#)
- DeselectColumn function [163](#)
- DeselectColumnsRange function [163](#)
- DeselectRow function [163](#)
- DeselectRowsRange function [163](#)
- DoubleClickCell function [163](#)
- Drag function (for JavaSlider) [163](#)
- Drag function (for JavaTable) [164](#)
- Exist function [169](#)
- Expand function [164](#)
- ExtendColumn function [165](#)
- ExtendColumnsRange function [165](#)
- ExtendRow function [165](#)
- ExtendRowsRange function [165](#)
- FireEvent function [164](#)
- FireEventEx function [164](#)
- GetAUTVar function [159, 169](#)
- GetProperty function [169](#)
- Insert function [165](#)
- InvokeMethod function [165](#)
- Maximize function [165](#)
- MethodWizard function [170](#)
- Minimize function [166](#)
- MouseDown function [166](#)
- Move function [166](#)
- Press function [166](#)
- QueryValue function [171](#)
- Resize function [166](#)
- Restore function [166](#)
- Select function (for JavaList) [166](#)
- Select function (for JavaMenu) [166](#)
- SelectCell function [166, 167](#)
- SelectCellRange function [167](#)
- SelectColumn function [167](#)
- SelectColumnRange function [167](#)
- SelectItem function [167](#)
- SelectRange function [167](#)
- SelectRow function [167](#)
- SelectRowsRange function [167](#)
- Set function [167](#)
- SetAUTVar function [159](#)
- SetInsertPos function [167](#)
- SetProperty function [171](#)
- UnSelect function [168](#)
- UnSelectRange function [168](#)



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Java applets
 recording and running tests on [155](#)
 testing [153–171](#)

Java applets or objects
 creating checkpoints [157](#)
 using scripting functions in tests [161](#)

Java objects, supported toolkits [154](#)

Java tab, Test Settings dialog box [488](#)

Java test settings, retrieving and setting [159](#)

Java testing options
 column_number [489](#)
 edit_replay_mode [490](#)
 event_model [490](#)
 max_column_gap [491](#)
 max_line_deviation [491](#)
 max_list_columns [491](#)
 max_row_gap [491](#)
 max_text_distance [491](#)
 record_by_number [492](#)
 table_record_mode [492](#)

JFC [154](#)

K

key assignments, creating [509](#)

keyboard shortcuts
 creating [509](#)
 deleting [509](#)
 editing [509](#)

L

Link URL check box [469](#)

Load ActiveX Controls check box [467](#)

Load Images check box [467](#)

Load Java Applets check box [467](#)

Load Time check box [469](#)

local data sheets [308](#)

local parameter, defined [197](#)

Location option [481](#)

Books
Online

Find

Find
Again

Help

Top of
Chapter

Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

M

Macromedia Flash objects

checking [176](#)ClipGotoFrame function [179](#)ClipGotoLabel function [179](#)ClipPlay function [179](#)ClipStop function [179](#)GetClipCurrentFrame function [179](#)GetClipCurrentLabel function [179](#)GetVariable function [179](#)GotoFrame function [179](#)Play function [179](#)ProportionalMove function [179](#)recording and running tests on [174](#)SetVariable function [180](#)Stop function [180](#)testing [174–180](#)Type function [178](#)using scripting functions [178](#)Main toolbar, Astra QuickTest window [29, 38](#)managing tests [63–64](#)creating new [63](#)opening [63](#)printing [64](#)saving [64](#)managing the testing process [25, 522–542](#)Mapped to Class option [487](#)mathematical formulas [323, 325](#)max_column_gap Java testing option [491](#)max_line_deviation Java testing option [491](#)max_list_columns Java testing option [491](#)max_row_gap Java testing option [491](#)max_text_distance Java testing option [491](#)Medium event recording configuration level [387](#)menu bar, Astra QuickTest window [29](#)Mercury Interactive on the Web [15](#)Mercury Tours sample application [14, 24, 253](#)Microsoft Excel [309](#)

Microsoft Query

choosing a database for a database

checkpoint [128–129, 322](#)database check [124–129](#)Microsoft Query, supported versions [128](#)Microsoft Visual Basic Scripting language [24](#)

modifying

default properties [65–73](#)descriptions [71–73](#)steps [54–56](#)multimedia applications, testing [172–190](#)multiple actions in a test [270–273](#)Books
Online

Find

Find
Again

Help

Top of
Chapter

Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

N

- Name option [481](#)
- Navigation Fallback Properties dialog box [483](#)
- Navigation Timeout box [483](#)
- New Action button [38](#), [287](#)
- New Action command [287](#)
- New button [37](#)
- New command [39](#)
- Number of Images check box [469](#)
- Number of Links check box [469](#)
- Numeric Content verification
 - databases [135](#)
- Numeric Range verification
 - databases [135](#)

O

- object descriptions, modifying [71–73](#)
- Object Mapping tab, Test Settings dialog box [485](#)
- object properties, viewing with the GUI Spy [66](#)
- Object Repository [72](#)
- Object Repository command [39](#)
- Object Repository dialog box [55](#), [249](#)
- Object Repository, deleting an object from [60](#)
- Object Repository, finding an object property or value [56](#)
- Object Repository, replacing an object property value [57](#)

- Object Timeout box [481](#)
- objects
 - dynamic descriptions [70](#)
 - Expert View [429](#)
 - identifying [65–73](#)
- ODBC
 - choosing a database for a database checkpoint [322](#)
- On Run Error box [464](#)
- Open Astra Test dialog box [48](#), [63](#)
- Open Astra Test from TestDirector Project dialog box [538](#)
- Open button [37](#), [48](#), [63](#)
- Open command [39](#)
- Open dialog box [63](#)
- Open New Web Browser Window at the Following URL option [477](#)
- opening tests [63](#)
 - in a TestDirector project [537](#)
- optional step [351–354](#)
- options
 - setting testing options for a single test [471–495](#)
 - setting testing options for all tests [458–470](#)
- Options dialog box
 - ActiveScreen tab [466](#)
 - General tab [463](#)
 - Page Verification tab [468](#)



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- options, testing
 - See setting testing options
 - Oracle [154](#)
 - Other Location option [481](#)
 - Output Parameter command [41](#)
 - Output Parameter Properties dialog box [239](#)
 - output parameters [220–245](#)
 - definition [221](#)
 - objects [236–241](#)
 - Output Parameter Properties dialog box [239](#)
 - Page Output Parameter Properties dialog box [225](#)
 - pages [223–227](#)
 - Table Output Parameter Properties dialog box [244](#)
 - tables [242–245](#)
 - text [228–235](#)
 - Text Output Parameter Properties dialog box [230](#)
 - Owner option [481](#)
- P**
- Page Checkpoint Properties dialog box [85, 86](#)
 - page checkpoints, automatic [83](#)
 - Page Output Parameter Properties dialog box [225](#)
 - Page properties
 - navigation fallback properties [483](#)
 - Page Verification tab, Options dialog box [468](#)
 - Parameter Info command [41](#)
 - Parameter object
 - default property [333](#)
 - functions and properties of [333](#)
 - Name property [334](#)
 - RawValue property [334](#)
 - Value property [333](#)
 - ValuebyRow property [334](#)
 - parameterization [191–219](#)
 - checkpoints [209–211](#)
 - example [212–219](#)
 - overview [192](#)
 - steps [198–206](#)
 - parameterization (output) [220–245](#)
 - objects [236–241](#)
 - Output Parameter Properties dialog box [239](#)
 - Page Output Parameter Properties dialog box [225](#)
 - pages [223–227](#)
 - Table Output Parameter Properties dialog box [244](#)
 - tables [242–245](#)
 - text [228–235](#)
 - Text Output Parameter Properties dialog box [230](#)



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

parameterized test, example [212–219](#)
 parameters, Expert View [426](#)
 Password Encoder dialog box [207](#)
 password encoding [207](#)
 Paste command [40](#)
 Paste command, data table [313](#)
 Paste Values command, data table [313](#)
 Pause button [38](#), [350](#), [375](#)
 Pause command [41](#)
 pausing test runs [375](#)
 planning tests [46](#), [??–71](#)
 Print button [37](#), [64](#), [371](#)
 Print command [39](#)
 Print command, data table [312](#)
 print options [506](#), [508](#)
 printing tests [64](#)
 programming
 adding functions to tests [405–411](#)
 comments [419](#)
 conditional statements [412](#)
 Expert View [420–449](#)
 Function wizard [404](#), [405–411](#)
 in the Expert View [442](#)
 sending messages to test results [417](#)
 Tree View [403–419](#)

project (TestDirector)
 connecting Astra QuickTest to a [529–533](#)
 disconnecting from a [532](#)
 opening tests in a [537](#)
 running tests from a [540](#)
 saving tests to a [534–536](#)
 Properties command [39](#)
 Properties tab, Test Settings dialog box [480](#)
 properties, default [65–73](#)

Q

query file for a database checkpoint, working
 with ODBC/Microsoft Query [322](#)
 query file, creating for a database checkpoint
[128–129](#)

R

Readme file [14](#)
 Real objects
 ProportionalClick function [188](#)



Books
Online



Find

Find
Again



Help



Top of
Chapter



Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- Real Player objects
 - checking [184](#)
 - Close function [188](#)
 - GetCurrentTime function [189](#)
 - OpenUrl function [188](#)
 - Pause function [188](#)
 - Play function [188](#)
 - recording and running tests on [181](#)
 - Seek function [189](#)
 - Stop function [189](#)
 - testing [181–190](#)
 - using scripting functions on [188](#)
 - WaitClipEnd function [189](#)
 - WaitClipTime function [189](#)
- Recalc command, data table [315](#)
- Record button [38](#)
- record_by_number Java testing option [492](#)
- Recording status options [399](#)
- recording tests [47–50](#)
- Redo command [40](#)
- regular expressions [246–267](#)
 - defining in object checkpoints [253](#)
 - defining in steps [248](#)
 - defining in text checkpoints [257](#)
 - overview [247](#)
 - syntax [261](#)
 - treating special characters literally [252, 256, 260](#)
- Rename Action command [302](#)
- Rename command [40](#)
- Replace command [40](#)
- Replace command, data table [314](#)
- Repository Editor [199](#)
- resetting standard event recording configuration settings [401](#)
- resources
 - Astra Function Reference [14](#)
 - Astra QuickTest help [14](#)
 - Astra QuickTest Readme file [14](#)
 - Astra QuickTest Tutorial [14](#)
 - books online [14](#)
 - Mercury Interactive on the Web [15](#)
 - support information [15](#)
 - technical support online [15](#)
 - VBScript reference guide [15](#)
- result set [124](#)
- Run All Iterations in Data Table option [285, 479](#)
- Run button [38, 349, 375](#)
- Run command [41](#)
- Run for Specified Iterations in Data Table option [286, 479](#)
- Run Mode box [464](#)
- Run Scripts check box [467](#)
- Run tab, Test Settings dialog box [477](#)
- Run Test dialog box [349](#)



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

running tests **346–357**
 from a TestDirector project **540**
 Pause button **350**
 Run button **349**
 Run Test dialog box **349**
 Stop button **350**
 viewing test results **362**

running transaction files
 Run Test dialog box **349**

Runtime Data table, Test Results window **361**

run-time settings, adding and removing **519**

S

sample application, Mercury Tours **14, 24, 253**

Save Astra Test dialog box **64**

Save Astra Test to TestDirector Project dialog box **535**

Save button **37, 64**

Save command **39**

Save dialog box **64**

saving tests **64**

saving tests to a TestDirector project server (TestDirector), disconnecting from a **533**

setting
 ActiveScreen options in the Options dialog box **466**
 ActiveScreen options in the Test Settings dialog box **494**
 breakpoints **376**
 General options in the Options dialog box **463**
 Java options in the Test Settings dialog box **488**
 Object Mapping options in the Test Settings dialog box **485**
 Page Verification options in the Options dialog box **468**
 Properties options in the Test Settings dialog box **480**
 Run options in the Test Settings dialog box **477**
 StartUp options in the Test Settings dialog box **475**
 Web options in the Test Settings dialog box **482**

Setting object **514**

setting testing options
See also testing options

setting testing options, within a test script **512–520**



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- Sheet object
 - AddParameter function [332](#)
 - DeleteParameter function [332](#)
 - functions and properties of [331](#)
 - GetCurrentRow function [332](#)
 - GetParameter function [331](#)
 - GetParameterCount function [331](#)
 - GetRowCount function [332](#)
 - Name property [331](#)
 - SetCurrentRow function [332](#)
 - SetNextRow function [332](#)
 - SetPrevRow function [332](#)
 - shortcut keys [39–42](#)
 - Show Action button [32](#)
 - Show Welcome Screen on Start check box [465](#)
 - Sort command, data table [315](#)
 - Specify SQL statement screen, for creating database checkpoints [127](#)
 - standard event recording configuration [387–388](#)
 - Start Record button [49](#)
 - Start Recording dialog box [49](#)
 - StartUp tab, Test Settings dialog box [475](#)
 - status bar, Astra QuickTest window [29](#)
 - Step commands [374](#)
 - Step Into button [38, 374](#)
 - Step Into command [41](#)
 - Step Out button [38, 374](#)
 - Step Out command [42](#)
 - Step Over button [38, 374](#)
 - Step Over command [41](#)
 - steps
 - modifying [54–56](#)
 - optional [351–354](#)
 - Stop button [38, 50, 350](#)
 - Stop command [41](#)
 - support information [15](#)
- T**
- Table Checkpoint Properties dialog box [117](#)
 - Table Output Parameter Properties dialog box [244](#)
 - table_record_mode Java testing option [492](#)
 - technical support online [15](#)
 - test
 - checking objects [108–114](#)
 - checking pages [83–97](#)
 - checking tables [115–120](#)
 - checking text [98–107](#)
 - checking Web objects [81–120](#)
 - debugging [372–383](#)
 - diagram [274, 290](#)
 - managing [63–64](#)
 - multiple actions in [270–273](#)



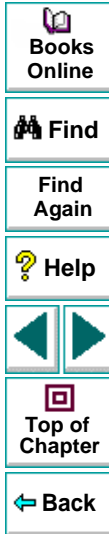
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- test (*cont'd*)
 - opening [63](#)
 - parameterization [191–219](#)
 - parameterization (output) [220–245](#)
 - pausing runs [350, 375](#)
 - planning [46, ??–71](#)
 - printing [64](#)
 - printing results [371](#)
 - programming [403–419](#)
 - recording [47–50](#)
 - running [346–357](#)
 - saving [64](#)
 - test results [358–371](#)
- Test Batch Runner [355–357](#)
- test batch, running a [355–357](#)
- test flow, definition [275](#)
- Test pane [29, 31](#)
 - Expert View tab [33](#)
 - Tree View tab [31](#)
- Test Results button [363](#)
- Test Results window [360](#)
 - Runtime Data table [361](#)
 - test results details [361](#)
 - test results tree [361](#)
- test results, sending messages to [417](#)
- test scripts
 - customizing [496–511](#)
 - highlighting script elements [503](#)
 - print options [506](#)
 - script window customization [508](#)
- Test Settings dialog box
 - ActiveScreen tab [494](#)
 - Java tab [488](#)
 - Object Mapping tab [485](#)
 - Properties tab [480](#)
 - Run tab [477](#)
 - StartUp tab [475](#)
 - Web tab [482](#)
- test tree
 - creating [50](#)
 - definition [52](#)
- test window
 - customizing appearance of [499](#)
 - highlighting script elements [503](#)
- TestDirector
 - managing the testing process [25](#)
 - modes of operation [525](#)
 - using Astra QuickTest with [25, 527](#)
 - working with [522–542](#)



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- TestDirector project
 - connecting Astra QuickTest to a [529–533](#)
 - disconnecting from a [532](#)
 - opening tests in a [537](#)
 - running tests from a [540](#)
 - saving tests to a [534–536](#)
 - TestDirector server [529–533](#)
 - connecting Astra QuickTest [529](#)
 - disconnecting from a [533](#)
 - testing in Expert View [420–449](#)
 - testing options
 - AutomaticLinkCheck [514](#)
 - DefaultLoadTime [514](#)
 - DefaultTimeOut [515](#)
 - restoring [518](#)
 - retrieving [517](#)
 - run-time [519](#)
 - setting [514](#)
 - setting for a single test [471–495](#)
 - setting for all tests [458–470](#)
 - WebTimeout [515](#)
 - within a test script [512–520](#)
 - testing process [20](#)
 - analyzing test results [23](#)
 - creating tests [20](#)
 - running tests [22](#)
 - Text Checkpoint Properties dialog box [100](#),
[258](#)
 - Text Output Parameter Properties dialog box
[230](#)
 - title bar, Astra QuickTest window [29](#)
 - Toggle Breakpoint button [376](#)
 - Toggle Breakpoint command [42](#)
 - Toggle Breakpoints button [38](#)
 - toolbars, Astra QuickTest window
 - Debug [29, 38](#)
 - File [29, 37](#)
 - Main [29, 38](#)
 - toolkits supported for java objects [154](#)
 - Tree View tab, Test pane [31](#)
 - tutorial [14](#)
 - typographical conventions in this guide [16](#)
- ## U
- Undo command [39](#)
 - Use Data Table Formula option [325](#)
 - Use Existing Web Browser Window option
[477](#)



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

V

- Variables tab, Debugger pane [36](#)
- Variables tab, Debugger view [379](#)
- VBScript
 - Astra Functions [14](#)
 - reference guide [15](#)
- verification method
 - for databases [135](#)
- verification type
 - for databases [134](#)
- View Results when Test Run Ends check box [464](#)
- viewing test results [358–371](#)
 - checkpoints [367](#)
 - filtering results [365](#)
 - printing test results [371](#)
 - Runtime Data table [369](#)
 - Test Results button [363](#)
 - Test Results window [360](#)

W

- Wait function [436](#)
- Watch Expressions tab, Debugger pane [36](#)
- Watch Expressions tab, Debugger view [379](#)
- Web event configuration
 - custom [389–400](#)
 - standard [387–388](#)
- Web event recording configuration [385–402](#)
- Web Event Recording Configuration dialog box [388](#)
- Web tab, Test Settings dialog box [482](#)
- WebTimeout testing option [515](#)
- Welcome to Astra QuickTest window [27](#)
- While statement, in the Expert View [447](#)

Books
Online

Find

Find
Again

Help

Top of
Chapter

Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Astra QuickTest 5.0 User's Guide

© Copyright 2000 by Mercury Interactive Corporation

All rights reserved. All text and figures included in this publication are the exclusive property of Mercury Interactive Corporation, and may not be copied, reproduced, or used in any way without the express permission in writing of Mercury Interactive. Information in this document is subject to change without notice and does not represent a commitment on the part of Mercury Interactive.

Mercury Interactive may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents except as expressly provided in any written license agreement from Mercury Interactive.

WinRunner, XRunner, LoadRunner, TestDirector, TestSuite, and WebTest are registered trademarks of Mercury Interactive Corporation in the United States and/or other countries. Astra SiteManager, Astra SiteTest, Astra QuickTest, Astra LoadTest, Topaz, RapidTest, QuickTest, Visual Testing, Action Tracker, Link Doctor, Change Viewer, Dynamic Scan, Fast Scan, and Visual Web Display are trademarks of Mercury Interactive Corporation in the United States and/or other countries.

This document also contains registered trademarks, trademarks and service marks that are owned by their respective companies or organizations. Mercury Interactive Corporation disclaims any responsibility for specifying which marks are owned by which companies or organizations.

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

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089
Tel. (408) 822-5200 (800) TEST-911
Fax. (408) 822-5300

AQTUG5.0/01

