

Astra LoadTest[®]™
Virtual User Recorder User's Guide
Version 5.4.3



MERCURY INTERACTIVE

Astra LoadaTest Virtual User Recorder User's Guide, Version 5.4.3

© Copyright 1994 - 2002 by Mercury Interactive Corporation

All rights reserved. All text and figures included in this publication are the exclusive property of Mercury Interactive Corporation or its licensors, 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.

Mercury Interactive and Design, M and Design, WinRunner, XRunner, LoadRunner, TestDirector, TestSuite, WebTest, Astra, Astra SiteManager, Astra SiteTest, SiteRunner, FreshWater Software, SiteScope, and SiteSeer are registered trademarks of Mercury Interactive Corporation in the United States and/or other countries. Astra QuickTest, Astra LoadTest, Astra FastTrack, RapidTest, QuickTest, Visual Testing, Action Tracker, Link Doctor, Change Viewer, Dynamic Scan, Fast Scan, Visual Web Display, ActiveTest, ActiveTest SecureCheck, ActiveWatch, POPs on Demand, Topaz, Topaz ActiveAgent, Topaz Observer, Topaz Prism, Topaz Delta, Topaz Rent-a-POP, Topaz Open DataSource, Topaz AIMS, Topaz Console, Topaz Diagnostics, Topaz WeatherMap, Twinlook, TurboLoad, LoadRunner TestCenter, SiteReliance and Global SiteReliance are trademarks of Mercury Interactive Corporation in the United States and/or other countries.

This document may also contain registered trademarks, trademarks, service marks and/or trade names 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@merc-int.com.

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089 USA

Table of Contents

Welcome to Astra LoadTest	ix
Using This Guide	ix
Online Resources	x
Typographical Conventions.....	xii

PART I: STARTING THE TESTING PROCESS

Chapter 1: Introduction	3
Testing with Astra LoadTest	3
Testing Process.....	4
Expert View	7
Actions.....	7
Sample Site	8
Managing the Testing Process	8
Chapter 2: The Virtual User Recorder at a Glance.....	9
Starting the Virtual User Recorder	9
The Virtual User Recorder Window	9
Test Pane.....	11
Display Pane	12
Data Pane.....	13
Using Virtual User Recorder Commands	13

PART II: CREATING TESTS

Chapter 3: Creating Tests	21
About Creating Tests	21
Planning a Test	22
Recording a Test	23
Creating Checkpoints.....	26
Understanding Your Test	27
Modifying Object Properties in Your Test	28
Deleting an Object from the Object Repository	33
Changing the ActiveScreen	34
Managing a Test	35
Chapter 4: Understanding Object Identification	37
About How the Virtual User Recorder Identifies Objects	37
Viewing Object Properties Using the Object Spy.....	38
Understanding How the Virtual User Recorder Learns Objects.....	40
Understanding Dynamic Descriptions of Objects	41
Modifying How the Virtual User Recorder Identifies Objects	41
Viewing Object Methods and Method Syntax.....	44
Chapter 5: Creating Checkpoints	47
About Creating Checkpoints.....	47
Checking Objects	48
Adding Checkpoints to a Test	48
Understanding the Checkpoint Properties Dialog Box	49
Modifying Checkpoints.....	51
Chapter 6: Checking Web Objects	53
About Checking Web Objects	53
Recording and Running Tests on Web Sites	54
Checking Pages	55
Checking Text	65
Checking Objects	70
Checking Tables	75
Chapter 7: Parameterizing Tests	79
About Parameterizing Tests	79
Setting Parameters as Global or Local	83
Parameterizing Steps	83
Parameterizing Checkpoints	89
Example of a Parameterized Test.....	91

Chapter 8: Testing ActiveX Controls and Java Applets	97
About Checking ActiveX Controls.....	97
Recording and Running Tests on ActiveX Controls	98
Inserting Functions with the ActiveX Wizard	100
Using Scripting Functions with ActiveX Controls.....	107
Testing Java Applets	107
Chapter 9: Testing Load	109
Inserting Transactions	109
Inserting Rendezvous Points	112
Setting Run-Time Options.....	114
Run-Time Settings	115
Sending Messages to Output	126
Chapter 10: Creating Output Parameters	129
About Creating Output Parameters.....	129
Creating Page Output Parameters	131
Creating Text Output Parameters	135
Creating Object Output Parameters.....	141
Creating Table Output Parameters.....	145
Chapter 11: Using Regular Expressions	149
About Regular Expressions	149
Using Regular Expressions in Steps.....	150
Using Regular Expressions in Object Checkpoints	153
Using Regular Expressions in Text Checkpoints.....	156
Regular Expression Syntax	159
Chapter 12: Working with Actions	165
About Working with Actions	166
Using Multiple Actions in a Test.....	166
Using Global and Action Data Sheets	167
Using the Action Toolbar	169
Creating New Actions.....	170
Inserting Existing Actions	172
Nesting Actions	177
Setting Action Properties.....	179
Removing Actions From a Test.....	184
Guidelines for Working with Actions	186

Chapter 13: Working with Data Tables	189
About Working with Data Tables.....	189
Global and Local Sheets	190
Editing the Data Table.....	191
Importing Data from a Database.....	196
Using Formulas in the Data Table.....	200
Using Data Table Scripting Methods.....	204
Chapter 14: Handling Unexpected Events and Errors	205
About Handling Unexpected Events and Errors	205
Changing the Status of Exceptions	207
Modifying Exceptions	208
Adding New Exceptions	210
Deleting Exceptions	211
Configuring Event Handling.....	211

PART III: RUNNING AND DEBUGGING TESTS

Chapter 15: Running Tests in Stand-Alone Mode	215
About Running Tests	215
Choosing a Run Mode.....	216
Running a Test	216
Using Optional Steps.....	221
Chapter 16: Analyzing Test Results in Stand-Alone Mode	225
About Analyzing Test Results.....	225
The Test Results Window	226
Viewing the Results of a Test Run	227
Viewing the Results of a Checkpoint	230
Viewing the Runtime Data Table for a Parameterized Test	232
Printing Test Results	233
Reporting Defects Detected During a Test Run.....	233
Chapter 17: Debugging Tests	235
About Debugging Tests	235
Using the Step Commands.....	236
Pausing Test Runs.....	237
Setting Breakpoints.....	237
Deleting Breakpoints	238
Using the Debugger Views	239
Example of Debugging a Test.....	241

PART IV: ADVANCED FEATURES

Chapter 18: Configuring Event Recording	245
About Configuring Event Recording.....	245
Selecting a Standard Event Recording Configuration.....	246
Customizing the Event Recording Configuration	248
Saving and Loading Custom Configuration Files	255
Resetting Standard Event Recording Configuration Settings	255
Chapter 19: Enhancing Your Tests with Programming.....	257
About Enhancing Your Tests with Programming	257
Inserting Functions	258
Using Conditional Statements	264
Sending Messages to Your Test Results	268
Adding Comments	269
Chapter 20: Testing in the Expert View.....	271
About Testing in the Expert View	271
Understanding the Expert View	272
Programming in the Expert View.....	276
Enhancing Tests with Comments, Calculations, and Control-Flow Statements.....	283
Accessing Runtime Object Properties and Methods	290
Chapter 21: Working with Astra LoadTest—for Power Users	295
Recording and Running Tests	295
Working with Dynamic Web Content.....	296
Advanced Web Issues	298
Test Maintenance	299
Testing Localized Applications.....	300
Load Testing Questions	300

PART V: CONFIGURING THE VIRTUAL USER RECORDER

Chapter 22: Setting the Virtual User Recorder Testing Options.....	305
About Setting the Virtual User Recorder Options.....	305
Setting the Virtual User Recorder Testing Options	306
Selecting The Virtual User Recorder Testing Options.....	307
Chapter 23: Setting Testing Options for a Single Test.....	311
About Setting Testing Options for a Single Test	311
Setting Testing Options for a Single Test	312
Testing Options for a Single Test	313
Tuning the Test Replay.....	322

Chapter 24: Customizing the Expert View	331
About Customizing Your Test in the Expert View	331
Setting Display Options	332
Personalizing Editing Commands	339
Chapter 25: Setting Testing Options from a Test Script	341
About Setting Testing Options from a Test Script	341
Setting Testing Options	342
Retrieving Testing Options	343
Controlling the Test Run	343
Adding and Removing Run-Time Settings	344

PART VI: WORKING WITH TESTDIRECTOR

Chapter 26: Working with TestDirector	349
About Working with TestDirector	349
Using Astra LoadTest with TestDirector	352
Connecting to and Disconnecting from a Project	354
Saving Tests to a Project	358
Opening Tests in a Project	360
Running Tests from TestDirector	361
Index	363

Welcome to Astra LoadTest

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

Using This Guide

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

It contains 6 parts:

Part I Starting the Testing Process

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

Part II Creating Tests

Describes how to create tests, insert checkpoints, assign parameters, set runtime settings, 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 LoadTest.**

Part V Configuring the Virtual User Recorder

Describes how to change Astra LoadTest'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 LoadTest interacts with TestDirector, Mercury Interactive's test management tool.

Online Resources

Astra LoadTest includes the following online resources:



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

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

Books Online displays the *Astra LoadTest User's Guide* in PDF format. Online books can be read and printed using Adobe Acrobat Reader 4.0. Check Mercury Interactive's Customer Support Web site for updates to Astra LoadTest 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 LoadTest Context Sensitive Help describes dialog boxes and toolbar buttons, and provides procedural information.

Astra LoadTest 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 Mercury Interactive's home page, its Customer Support web 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, its products and services. This includes new software releases, seminars and trade shows, customer support, training, 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.
<i>Italics</i>	<i>Italic</i> 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.

Part I

Starting the Testing Process

1

Introduction

Welcome to Astra LoadTest, Mercury Interactive's load testing tool for Web applications.

Astra LoadTest allows you to test your Web site by running multiple virtual users (Vusers) on a workstation. With Astra LoadTest, you can effectively test your entire system architecture under various load conditions.

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

Testing with Astra LoadTest

Astra LoadTest facilitates creating tests on your Web application by recording as you navigate. You record your test with the Virtual User Recorder. As you navigate through your site, the Virtual User Recorder 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 LoadTest to check the properties of specific objects in your site. For example, you can instruct Astra LoadTest 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.

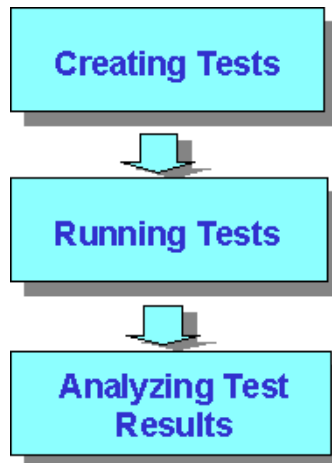
After you record, you can further enhance your test by adding and modifying steps in the test tree. When you run the test, Astra LoadTest connects to your site and performs each step in your test. After you run your test, you can view a report detailing which steps in your test succeeded or failed.

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 12, “Working with Actions.”

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

Testing Process

Testing with Astra LoadTest involves 3 main stages:



Creating Tests

You create a test by recording a Web session with the Virtual User Recorder. The test is used to load test your application.

To create a test:

- Record a session on your site.

As you navigate through your site, the Virtual User Recorder 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.”

- ▶ Insert load testing elements into your test.

You define *transactions* to mark the business processes that Astra LoadTest should measure. When you record a test, Astra LoadTest automatically marks each step you perform as a transaction. This means that when you run a load testing scenario, each step in your test tree is recognized as a transaction to be measured.

You insert *rendezvous points* into a test to emulate heavy user load on the server. Rendezvous points allow you to select specific steps in your test and regulate the load they are subjected to. They are used to check high usage areas such as log in screens or form submissions. For more information, see Chapter 9, “Testing Load.”

- ▶ 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 LoadTest substitutes the parameters in your test with values from the table. During each *iteration* of your test, Astra LoadTest changes the values in the parameterized statements. For more information, see Chapter 7, “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 10, “Creating Output Parameters.”

Running Tests

After you create your test, you run it using the Virtual User Recorder to debug it before you incorporate it into a load testing scenario.

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 LoadTest 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 LoadTest repeats the test for each set of data values you defined. For more information, see Chapter 15, "Running Tests in Stand-Alone Mode."

- ▶ 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 17, "Debugging Tests."

Note: Astra LoadTest runs the HTML scripts on the client side and supports both JavaScript and VBScript. This allows you to run your test without correlating it first.

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 16, “Analyzing Test Results in Stand-Alone Mode.”

- ▶ 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 26, “Working with TestDirector.”

Expert View

You can use the Expert View tab to view a text-based version of your test. The test script is composed of VBScript statements (Microsoft’s Visual Basic Scripting language) that correspond to the steps and checks displayed in your test tree. For more information, see Chapter 20, “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 7, “Parameterizing Tests.” For more information on working with actions, see Chapter 12, “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 LoadTest 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 LoadTest, you can create and save tests directly to your TestDirector project. You can also run Astra LoadTest tests from TestDirector and then use TestDirector to review and manage the results. For more information, see Chapter 26, “Working with TestDirector.”

2

The Virtual User Recorder at a Glance

This chapter explains how to start the Virtual User Recorder and introduces the Virtual User Recorder window.

This chapter describes:

- ▶ Starting the Virtual User Recorder
- ▶ The Virtual User Recorder Window
- ▶ Test Pane
- ▶ Display Pane
- ▶ Data Pane
- ▶ Using Virtual User Recorder Commands

Starting the Virtual User Recorder



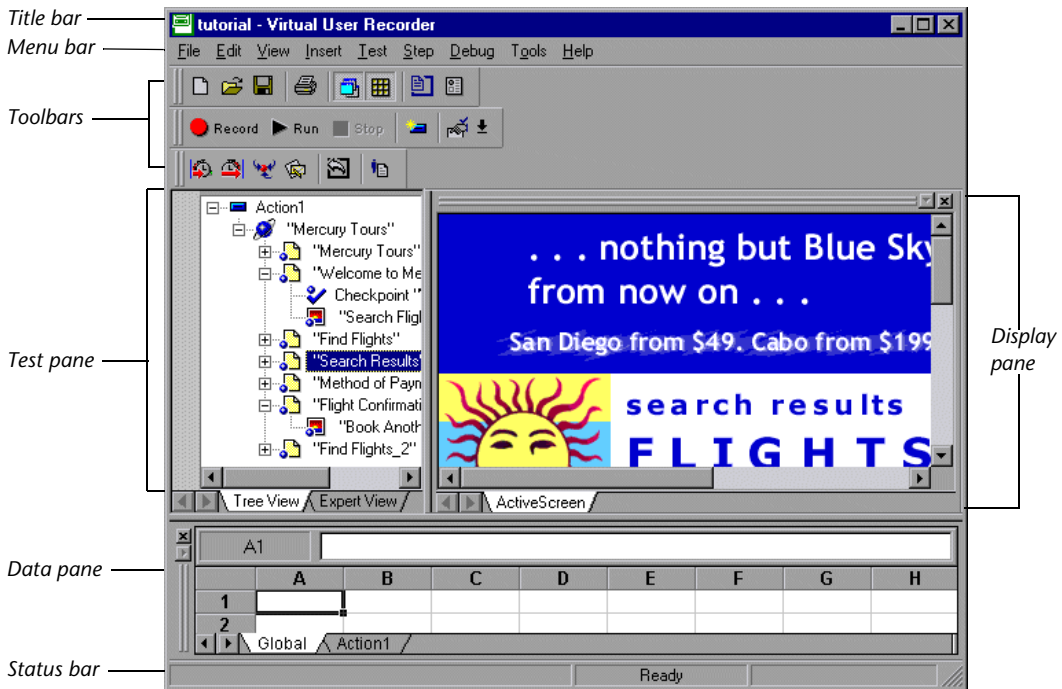
To start the Virtual User Recorder, click the Astra LoadTest icon from your desktop. After several seconds, the Welcome to Astra LoadTest screen opens. Select the Recorder icon to open the Virtual User Recorder.

The Virtual User Recorder Window

The Virtual User Recorder window contains the following key elements:

- ▶ *Title bar*, displaying the name of the currently open test
- ▶ *Menu bar*, displaying menus of the Virtual User Recorder 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 (only available when replaying tests)
- *Load Toolbar*, containing buttons to assist you in customizing your test.
- *Test pane*, containing two tabs to view your test—Tree View and Expert View
- *Display pane*, containing the ActiveScreen tab to assist you in the testing process
- *Data pane*, containing two tabs to assist you in parameterizing your test—Global and Action
- *Status bar*, displaying the status of the open test

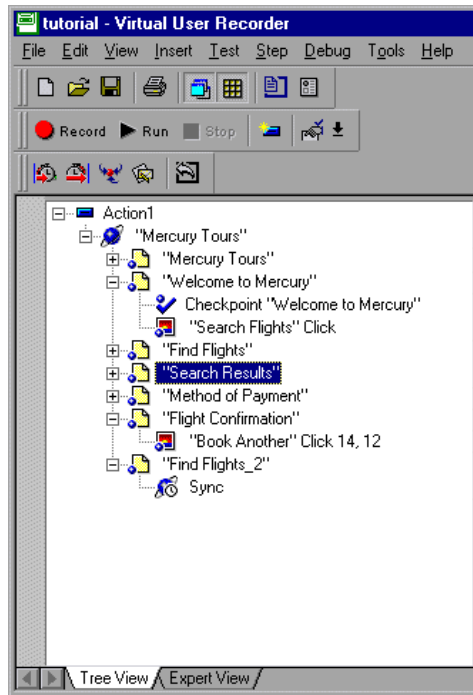


Test Pane

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

Tree View Tab

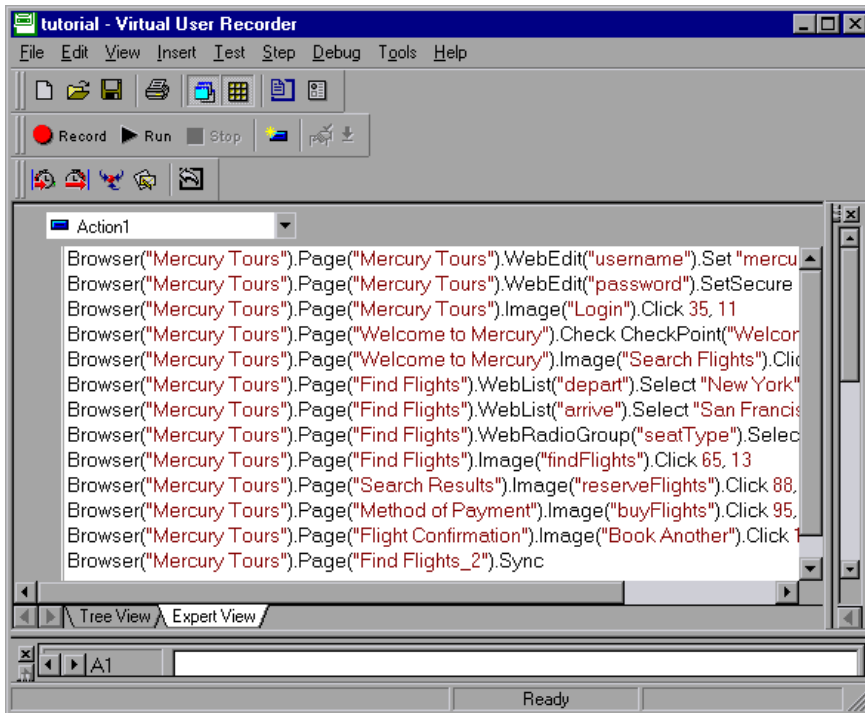
In the Tree View tab (default mode), the Virtual User Recorder displays your test in the form of a collapsible icon-based *test tree*. Each operation performed on your Web application is recorded as an icon in your test tree. For every icon in the Tree View, the Virtual User Recorder displays a corresponding line of script in the Expert View.



Expert View Tab

In the Expert View tab, the Virtual User Recorder displays your test in the form of a test script instead of a test tree. Your test 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 20, “Testing in the Expert View.”



Display Pane



Display Views Tab

The Display Views tab displays the Web 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



The Data pane contains two tabs to assist you in parameterizing your test—Global and Action. 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 LoadTest inserts the data from the Global tab into the test. For more information, see Chapter 12, “Working with Actions.”

Action Tab

The Action tab contains variable values for the parameters defined in your parameterized test. The variable values are available only for a specific action and not for an entire test. When you run your parameterized test, Astra LoadTest inserts the data from the Action tab into the relevant action in the test. For more information, see Chapter 12, “Working with Actions.”

Using Virtual User Recorder Commands

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

Choosing Commands from a Menu

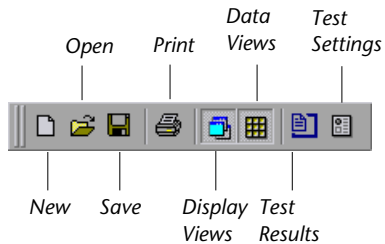
You can choose all Virtual User Recorder commands from the menu bar.

Clicking Commands on a Toolbar

You can execute some Virtual User Recorder commands by clicking buttons on the toolbars. The Virtual User Recorder has four built-in toolbars: the *File toolbar*, the *Main toolbar*, the *Load Toolbar* and the *Debug toolbar*. The Debug Toolbar is only available when replaying a test.

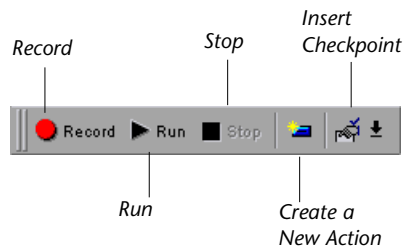
File Toolbar

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:



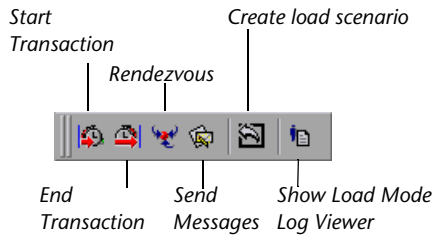
Main Toolbar

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



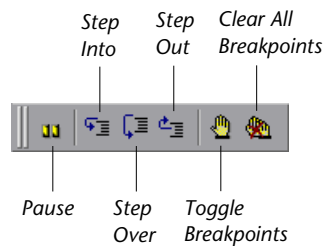
Load Toolbar

The Load toolbar contains buttons for the commands used when inserting transactions and rendezvous points into your test. You can also invoke the Controller when you are ready to run your scenario. The following buttons appear on the Load 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:



Note: The debug toolbar is automatically enabled when you run your test in the Virtual User Recorder.

Executing Commands Using Shortcut Keys

You can execute some Virtual User Recorder 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.
Export to Zip File	Ctrl + Alt + S	Exports the test to a zip file.
Import from Zip File	Ctrl + Alt + O	Imports a test from a zip file.
Print	Ctrl + P	Prints the active test.
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.
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).
Parameter Info	Ctrl + Shift + Space	Displays the syntax of a parameter (Expert View only)

Command	Shortcut Key	Function
Rename	F2	Changes the name of an action or a step (Tree View only).
Checkpoint	F12	Creates a checkpoint for a text string, an object, or a table.
Output Parameter	Ctrl + F12	Creates an output parameter for a text string, an object, or a table.
Record	F3	Starts the test recording.
Stop	F4	Stops test recording or the test run.
Run	F5	Runs the test from the beginning.
Object Properties	Ctrl + Enter	Opens the Object tab in the Parameterization/Properties dialog box.
Function Arguments	Alt + Enter	Opens the Method tab in the Parameterization/Properties dialog box.
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, however, if the current line calls a test or function, the called test or 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 another test or a function, the called test or function is executed in its entirety but is not displayed in the view.

Command	Shortcut Key	Function
Step Out	Shift + F11	Runs to the end of the called test or function, returns to the calling test, and then pauses execution. (Available only after entering a test or function using Step Into.)
Insert/Remove Breakpoint	F9	Sets or clears a breakpoint in the test.
Clear All Breakpoints	Ctrl + Shift + F9	Deletes all breakpoints in the test.

Part II

Creating Tests

3

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
- Deleting an Object from the Object Repository
- Changing the ActiveScreen
- Managing a Test

About Creating Tests

The Virtual User Recorder 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, the Virtual User Recorder 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, the Virtual User Recorder substitutes the parameters in your test with values from the table. During each *iteration*, or repetition, of your test, the Virtual User Recorder inserts a different value in the parameterized statements. The Virtual User Recorder 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 steps you want to record to create your test. Realistic tests that check specific functions and load performance of the application are best.
- ▶ Decide which information you want to check during the test. A checkpoint can check for differences in the text strings and objects 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 the Virtual User Recorder generally records by default, you can configure the events you want to record. For more information, see Chapter 18, “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 7, “Parameterizing Tests.”

- ▶ You can change the way that the Virtual User Recorder 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 Object Identification.”
- ▶ If you are an advanced user, consider using actions to streamline the testing process. For additional information, see Chapter 12, “Working with Actions.”

Recording a Test

You create a test by recording the typical processes that users perform. The Virtual User Recorder 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 12, “Working with Actions.”

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 the Virtual User Recorder records and displays your tests by setting testing options in the Options dialog box. For more information, see Chapter 22, “Setting the Virtual User Recorder Testing Options.”

To record a test:



- 1** Click the Astra LoadTest desktop icon to open the Welcome to Astra LoadTest screen. Select the Recorder icon to open the Virtual User Recorder.
- 2** Open a test:

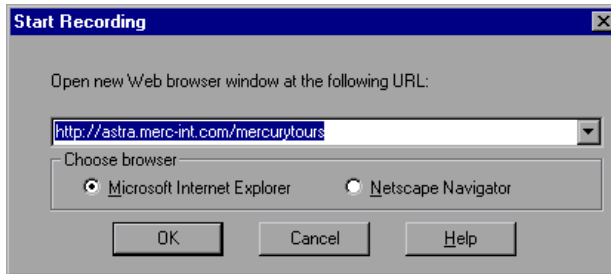


- To create a new test, click the **New** button or choose **File > New**.
- To open an existing test, click the **Open** button or choose **File > Open**. In the **Open Astra Test** dialog box, select a test and click **Open**.

For more information, see Managing a Test.



- 3 Click the **Record** button or choose **Test > Record**. The Start Recording dialog box opens.



- If this is your first recording session in a test, choose which browser to use and specify a URL. By default, the URL for the Mercury Tours site appears as the address.
- If this is not your first recording session in a test, the Virtual User Recorder remembers your choices from the previous session. Proceed to step 4.

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

The options you select in the Startup dialog box also set the startup settings for the test. For more information about startup settings, see “Browser Testing Options”.

- 4 Navigate through your Web site. The Virtual User Recorder records each step you make in the test tree in the Tree View tab.
- 5 You can insert text checkpoints, 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 insert load testing elements to measure how your application functions under load. For more information, see Chapter 9, “Testing Load.”
- 7 You can parameterize your test to check how it performs the same operations with multiple sets of data. For more information, see Chapter 7, “Parameterizing Tests.”



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



- 9 To save your test, click the **Save** button or choose **File > Save**. Assign a name to the test. For more information, see “Managing a Test”.
- 10 To save the recording as a zipped test for future importation into the Virtual User Recorder, you must zip the file using **File > Export to Zip File**. For more information, see “Managing a Test”.

Creating Checkpoints

You use the Virtual User Recorder to add checkpoints to your automated test. A *checkpoint* is a verification point on a Web page 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 .

You can check Web page statistics, text strings, and objects, and tables. You can also use formulas to check that the data displayed on a Web page is valid. For information, see Chapter 6, "Checking Web Objects."

For general information about creating checkpoints, see Chapter 5, "Creating Checkpoints."

Understanding Your Test

While recording, the Virtual User Recorder 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.
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>35, 9</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 LoadTest can continue to identify it. You can modify the object by modifying one or more of the object's property values in the Object Properties dialog box. For additional information about working with the object repository, see Chapter 4, "Understanding Object Identification."

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 LoadTest will continue to identify the link properly.

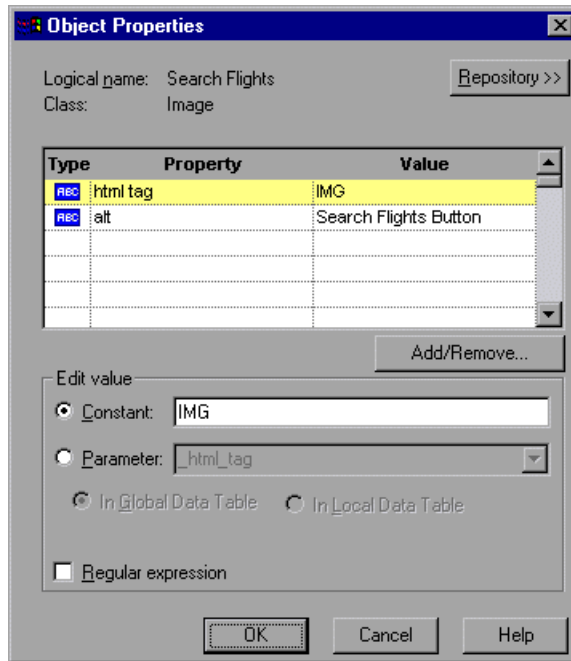
Both the Object Properties dialog box and the Object Repository dialog box identify objects on a per action basis. Thus, if the same object occurs in several steps within the same action, you need to modify the object's properties only one time. If the object occurs again in another action, however, you need to update the object's properties in that action as well. For more information about actions, see Chapter 12, "Working with Actions."

When using the Object Repository dialog box you only have access to the object you select. Using the Object Repository displays the object tree and enables you to modify any object in an action.

To modify an object in the Object Properties dialog box:

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

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



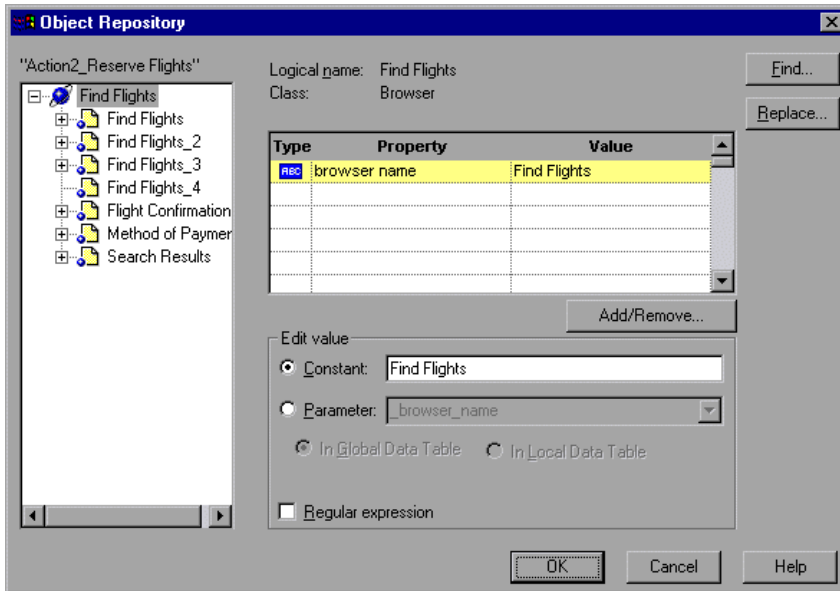
- 2 Highlight the property and value to modify.
- 3 In the **Constant** box, enter a new value for the property.
- 4 Click **OK** to close the dialog box.

To modify an object using the Object Repository:

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

Tip: You can open the Object Repository from the Object Properties dialog box by clicking the **Repository** button. The Object Repository dialog box opens and displays the test tree of the action.

The Object Repository dialog box opens and displays the test tree.



- 1 Select a step containing the object you want to modify.
- 2 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.

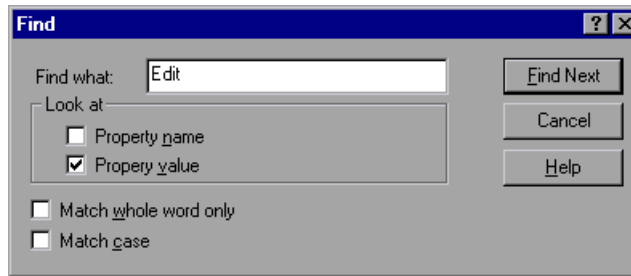
You can also use the **Find** and **Replace** buttons in the Object Repository dialog box to find and/or modify a property or value that occurs several times in the same action.

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 **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.



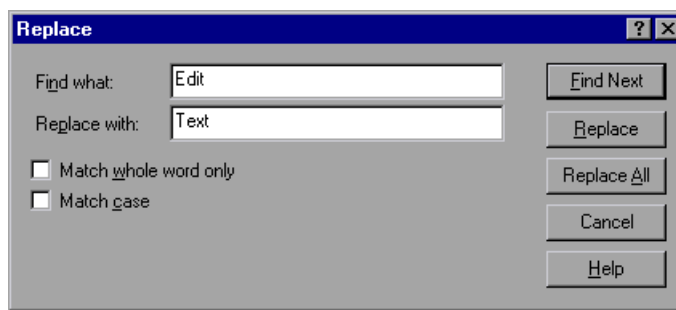
- 3 Enter the text for the property or value you want to find.
- 4 Select **Property name**, **Property value**, or both.
- 5 If you want the search to find only words that exactly match the text you entered, select **Match whole word only**.
- 6 If you want the search to distinguish between upper and 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 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 for the selected object opens and displays the properties Astra LoadTest uses to identify the object.

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



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

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

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

Deleting an Object from the Object Repository

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

Note: If your action contains references to an object that you have deleted from 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 LoadTest uses to identify the object.

- 2** In the test tree, right-click the object you want to delete and click **Delete**. A confirmation message appears.
- 3** Click **Yes** to confirm that you want to delete the object. The object is deleted from the object repository.

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

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 the Virtual User Recorder can continue to find the objects in your modified site.

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

To change the ActiveScreen:

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

Managing a Test

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

Creating a New Test



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

Opening an Existing Test

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

To open an existing test:



- 1** Click the **Open** button or choose **File > Open**. The Open 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 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**. The Virtual User Recorder 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.

Zipping a Test

While you record, the Virtual User Recorder creates a series of configuration, data, and source code files. These files contain run-time and setup information. The Virtual User Recorder saves these files together with the test. You can zip these files to conserve space and make the tests easier to transfer.

To zip a test:

- 1** Choose **File > Export to Zip File**. The Export to Zip File dialog box opens.
- 2** Type a zip file name and path, or accept the default name and path, and click **OK**. The recorderAstra LoadTest zips the test and its associated files.

Unzipping a Test

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

To unzip a zipped test:

- 1** Select **File > Import from Zip File**. The Import from Zip File dialog box opens.
- 2** Type or select the zip file that you want to unzip, choose a target folder into which you want to unzip the files, and click **OK**. The Virtual User Recorder unzips the test and its associated files.

Printing a Test

You can print your test.

To print a test:



- 1** Click the **Print** button or choose **File > Print**. The Print dialog box opens.
- 2** Click **OK** to print.

4

Understanding Object Identification

This chapter explains how the Virtual User Recorder identifies objects in your application and how to modify the way it identifies an object, which is useful when working with objects that change dynamically. It also describes how to view the available methods for an object, as well as the corresponding syntax, so that you can easily add statements to your script in the Expert View.

This chapter describes:

- ▶ Viewing Object Properties Using the Object Spy
- ▶ Understanding How the Virtual User Recorder Learns Objects
- ▶ Understanding Dynamic Descriptions of Objects
- ▶ Modifying How the Virtual User Recorder Identifies Objects
- ▶ Viewing Object Methods and Method Syntax

About How the Virtual User Recorder Identifies Objects

The Virtual User Recorder 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). You can view the properties of any object on your desktop using the Object Spy, and you can change the properties that The Virtual User Recorder uses to identify an object in the Test Object Repository.

Viewing Object Properties Using the Object Spy

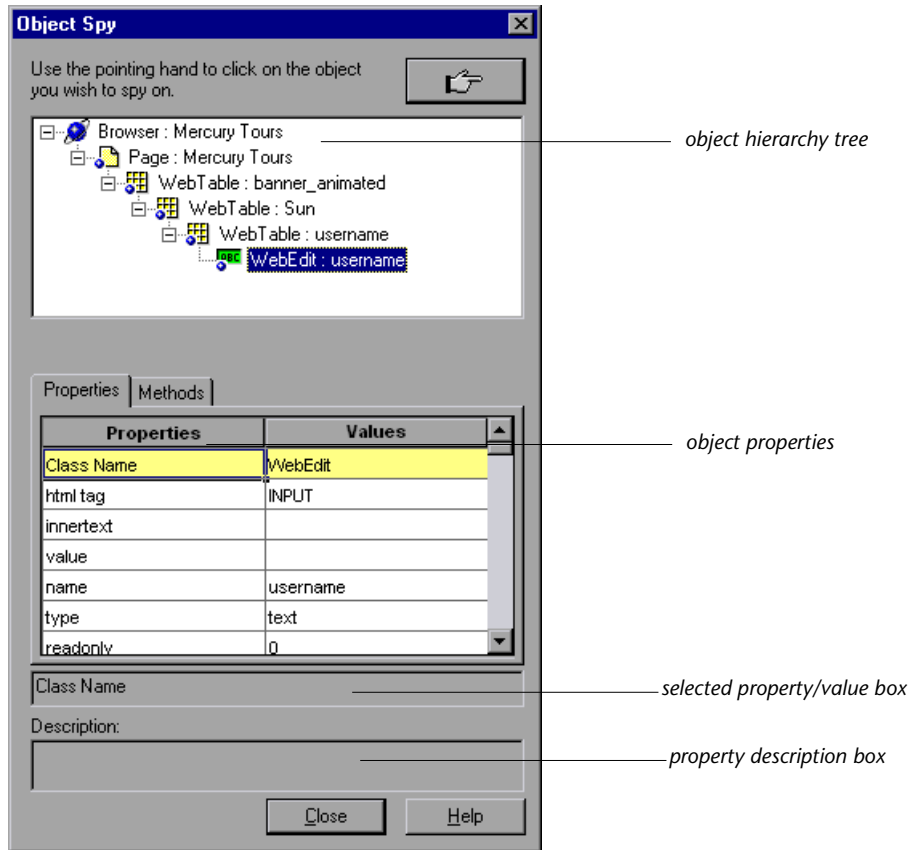
Using the Object Spy, you can view the properties of any object on your desktop. You use the Object Spy pointer to point to an object, and the Properties tab of the Object Spy displays the object hierarchy tree and the properties and values of the selected object in the Object Spy dialog box.

To view object properties:

- 1** Open your browser or application to the page containing the object on which you want to spy.
- 2** Choose **Tools > Object Spy** to open the Object Spy dialog box.
- 3** Click the pointing hand. Both the Virtual User Recorder and the Object Spy are minimized so that you can point to any object on the open application.



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



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

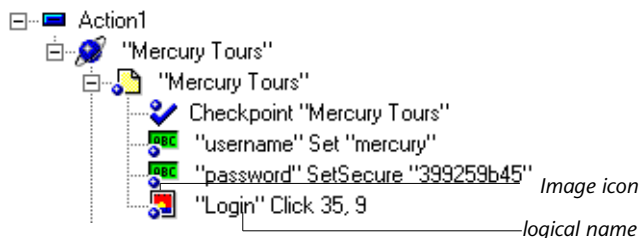
Note: If the value of a property contains more than one line, the value cell of the object properties list indicates: "multi-line value". To view the value, click the value cell. The selected property/value box displays the value with delimiters indicating the line breaks.

Understanding How the Virtual User Recorder Learns Objects

The Virtual User Recorder 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.

Note: To view the default properties of an object, right click the object in the test tree and select Object Properties.

For each object class, the Virtual User Recorder learns a set of default properties, which it uses to identify objects of that class in your application. For example, by default, the Virtual User Recorder 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 the Virtual User Recorder uses to identify an object 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.



Understanding Dynamic Descriptions of Objects

You can change the properties that the Virtual User Recorder uses to identify an object. This is useful when you want to create and run tests on an object that changes dynamically. An object may change dynamically if it is frequently updated or if it is created using dynamic content, e.g. from a database. You can also change the properties that identify an object in order to reference objects using properties that were not automatically learned while recording.

For example, suppose you are testing a Web site that contains an archive of news letters. The archive page includes a hypertext link to the current news letter and to all past news letters. The text in the first hypertext link on the page changes as the current news letter changes, but it always links to a page called *current.html*. Suppose you want to create a step in your test in which you always click the first hypertext link in your archive page. Since the news is always changing, the text in the hypertext link keeps changing. You need to modify how the Virtual User Recorder 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 the Virtual User Recorder Identifies Objects

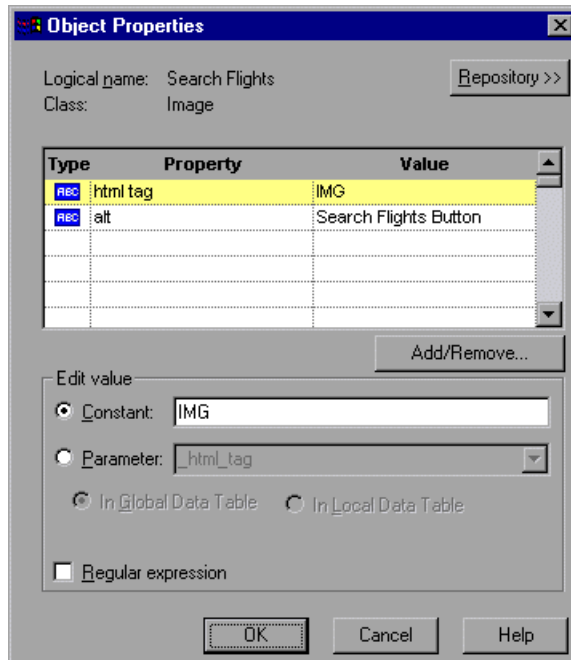
Suppose you are testing a Web page that contains an image that is an advertisement. Clicking this image opens the advertiser’s Web page. You do not want to identify the image by its name, since the image changes whenever the advertiser changes. Therefore, the value of the name property changes. You would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

The Object Properties dialog box contains the properties of a given object. Alternatively, the Object Repository contains a list of all objects in an action, their properties, and their values. For information about actions, see Chapter 12, "Working with Actions." By default, the Virtual User Recorder learns certain properties for each type of object. You can use the Add/Remove Properties dialog box to instruct the Virtual User Recorder to learn properties of the object other than the default properties.

To modify how the Virtual User Recorder identifies an object:

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

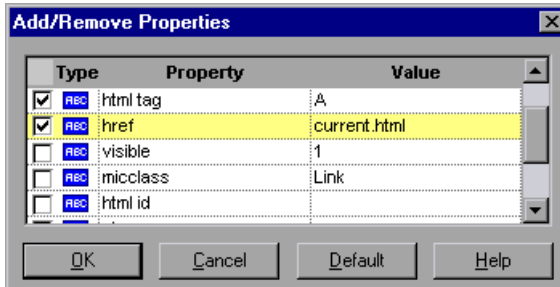
The Object Properties dialog box opens.



- 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 the Virtual User Recorder uses to identify the object. The value for each property is displayed in the Value column.



- 3 Modify the default properties that the Virtual User Recorder 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 Properties dialog box is reactivated.

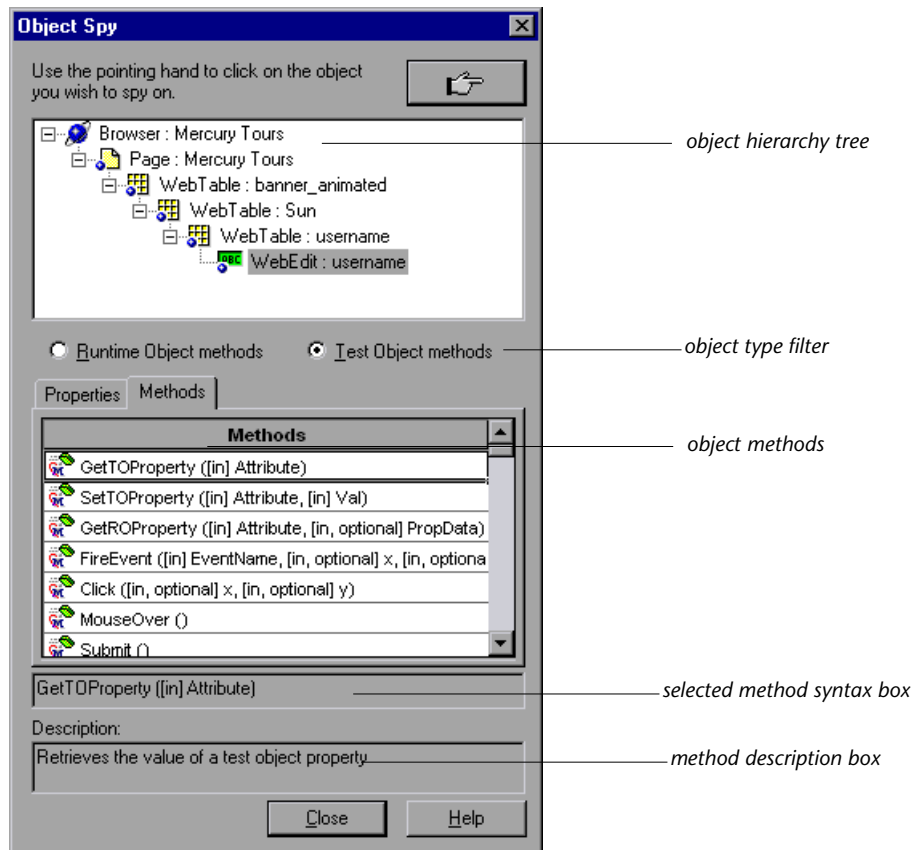
Tip: After you add a new property, you can modify its value in the Edit Value box in the Object Properties dialog box. For more information on modifying object properties, see Chapter 3, “Creating Tests.”

Viewing Object Methods and Method Syntax

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

To view object methods:

- 1 Open your browser or application to the page containing the object on which you want to spy.
- 2 Choose **Tools > Object Spy** to open the Object Spy dialog box.
- 3 Click the Methods tab.
- 4 Click the pointing hand. Both the Virtual User Recorder and the Object Spy are minimized so that you can point to any object on the open application.
- 5 Click the object for which you want to view the associated methods. The Object Spy returns to focus and displays the object hierarchy tree and the *test object* methods associated with the object that is selected within the tree.



- 6** To view the methods of the *runtime object*, click the **Object view** radio button.
- 7** If you want to view methods for another object within the displayed tree, click the object on the tree.
- 8** If you want to copy the syntax of a method to the clipboard, click the method in the list. The syntax is displayed in the selected method syntax box. Highlight the text and use CTRL-C to copy the text to the clipboard.

5

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

The Virtual User Recorder 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, the Virtual User Recorder adds a checkpoint icon under the highlighted step in the test tree. When you run the test, Astra LoadTest 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 check Web page statistics, text strings, and objects, and tables. You can also use formulas to check that the data displayed on a Web page is valid. For additional information, see Chapter 6, "Checking Web Objects."

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.

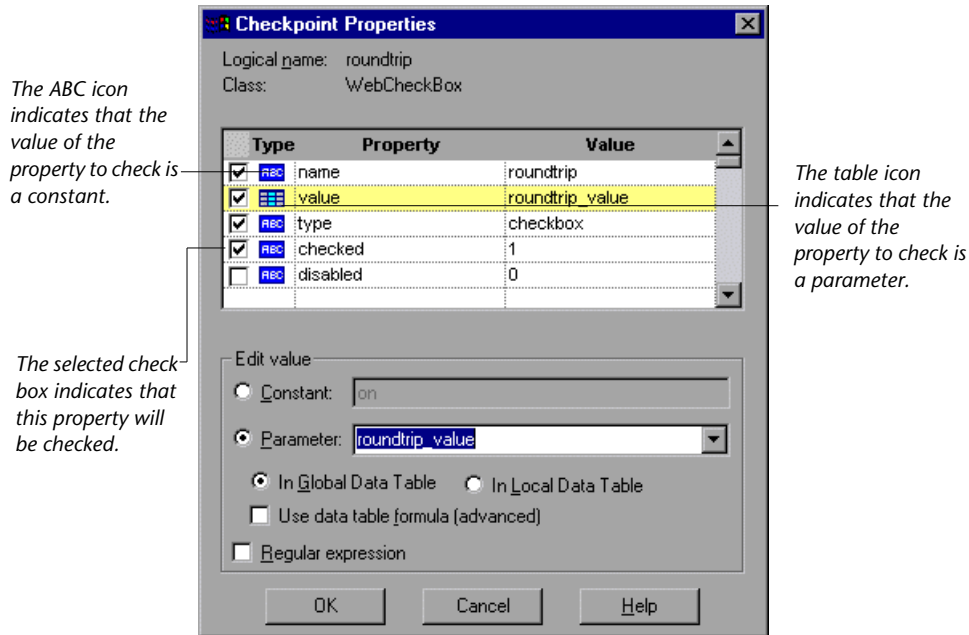
There are several ways to add checkpoints:



- ▶ 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.

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 Object Identification."

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 7, "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 on data tables, see Chapter 13, "Working with Data Tables." For more information on regular expressions, see Chapter 11, "Using Regular Expressions."

Modifying Checkpoints

If you already created a checkpoint 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** Double-click an existing checkpoint in the test tree. A checkpoint dialog box opens.
- 2** Modify the properties and click **OK**.

6

Checking Web Objects

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

This chapter describes:

- ▶ Recording and Running Tests on Web Sites
- ▶ Checking Pages
- ▶ Checking Text
- ▶ Checking Objects
- ▶ Checking Tables

About Checking Web Objects

You can check the Web objects in your Web site to determine whether 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, and other Web objects such as images and form elements.

Recording and Running Tests on Web Sites

You record on Web browsers to create tests to check Web objects. The Virtual User Recorder supports recording and running tests on the following Web browsers:

- Netscape Navigator
- Microsoft Internet Explorer

The Virtual User Recorder tests are cross-browser: you can record a test on one browser and run it on any other browser.

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



Checking Pages

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




Creating Page Checkpoints

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 Navigate to a page where you want to add a checkpoint.
- 2 Choose **Insert > Checkpoint** and click in the page as displayed by the browser. 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 57.
- 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.
-  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 57.
- 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.

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

Understanding the Page Checkpoint Properties Dialog Box

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



Identifying the Object

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

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

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, the Virtual User Recorder 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>

Pane Element	Description
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 Page Property," on page 58.

Editing the Value of a Page Property

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

Option	Description
Constant (default)	Sets the value of the property.
Parameter	Sets the property value as a parameter. For more information, see Chapter 7, "Parameterizing Tests."
In Global Data Table	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 12, "Working with Actions."
In Local Data Table	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 12, "Working with Actions."
Regular expression	Sets the value as a regular expression. For more information, see Chapter 11, "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 .

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

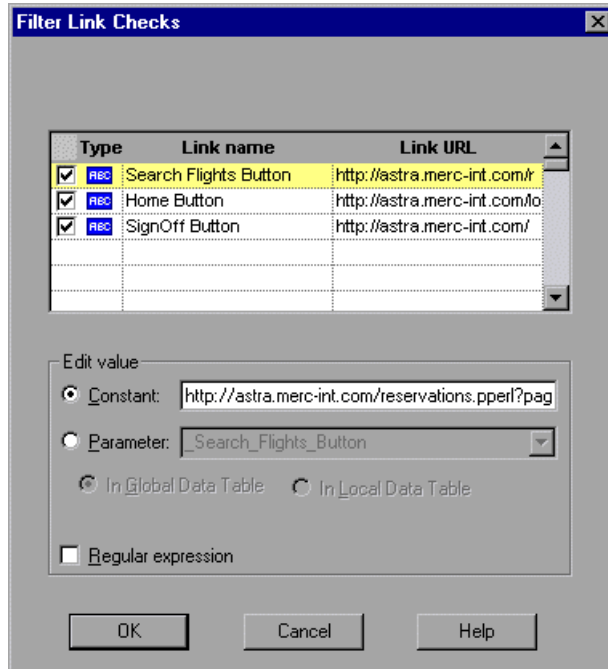
Checking All the Objects in a Page

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

Option	Description
Links	Checks the functionality of the links in the page according to your selections in the Filter Link Checks dialog box.
Filter Link Check	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 61.
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 63.
Broken Links	Instructs Astra LoadTest to check only for broken links that are targeted to your current host.



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 60.



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.

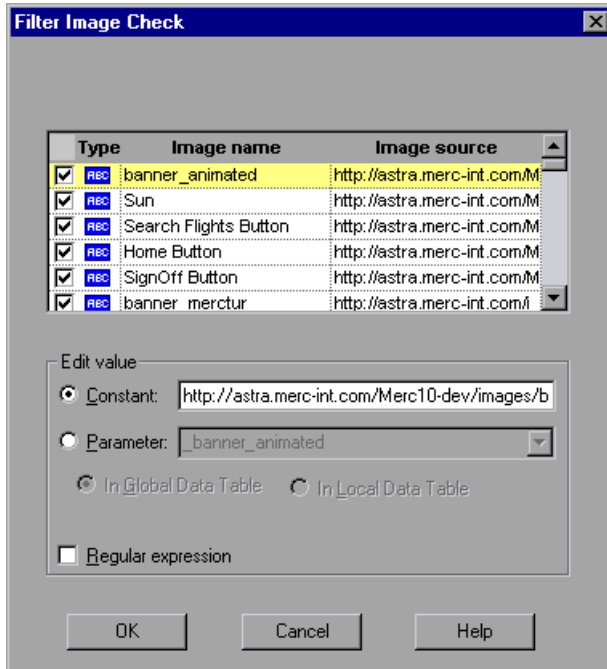
Editing the Value of the Target URL

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 7, "Parameterizing Tests."
In Global Data Table	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 12, "Working with Actions."
In Local Data Table	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 12, "Working with Actions."
Regular expression	Sets the value as a regular expression. For more information, see Chapter 11, "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 60.



Choosing which Image Sources to Check

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

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

Editing the Value of the Path of the Image Source File

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

Option	Description
Constant (default)	Sets the 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 7, "Parameterizing Tests."
In Global Data Table	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 12, "Working with Actions."

Option	Description
In Local Data Table	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 12, “Working with Actions.”
Regular expression	Sets the value as a regular expression. For more information, see Chapter 11, “Using Regular Expressions.”

Checking Text

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

Creating a Text Checkpoint

You can add a text checkpoint while recording or afterward.

To add a text checkpoint while recording:

1 Highlight a text string on the Web page.



2 Choose **Insert > 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 67.

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.

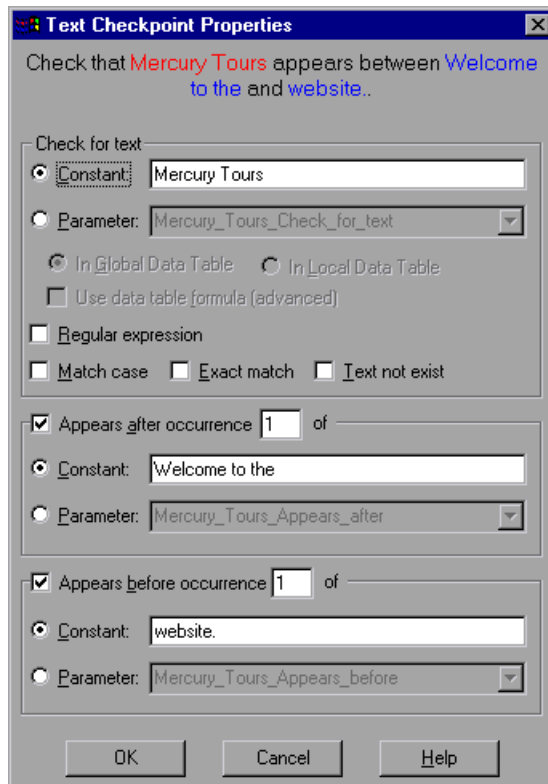
The ActiveScreen displays the Web page corresponding to the highlighted step.

- 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 67.
- 6** Click **OK** to close the dialog box.

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

Understanding the Text Checkpoint Properties Dialog Box

In the Text Checkpoint Properties dialog box, you can specify the checked text as well as which text is displayed before and after the checked text. This is particularly helpful when the text string you want to check is displayed several times in the same Web page. For example, you want to check a text string that is displayed several times in a Web page. Each time it appears it is preceded and followed by the same text. Suppose you want to check the third occurrence of the text string. By specifying the third occurrence of the preceding and following text, you can check the exact text you are interested in.



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 the Virtual User Recorder checks when running the test.
Parameter	Sets the text string as a parameter. For more information, see Chapter 7, "Parameterizing Tests."
In Global Data Table	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 12, "Working with Actions."
In Local Data Table	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 12, "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. The Virtual User Recorder 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 13, "Working with Data Tables."
Regular expression	Sets the text string as a regular expression. For more information, see Chapter 11, "Using Regular Expressions."
Match case	Conducts a case sensitive search.
Exact match	Checks according to the exact expected text.
Text not exist	Checks that the text string does not appear.

Specifying What Appears After the Text to Check

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

Option	Description
Appears after occurrence _ of (default)	Checks that the text to check appears after the specific occurrence of the text in this box. To ignore the text that appears after the text to check, clear this check box.
Constant (default)	Displays the text that appears after the text to check.
Parameter	Sets the text string as a parameter. For more information, see Chapter 7, “Parameterizing Tests.”

Specifying What Appears Before the Text to Check

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

Option	Description
Appears before occurrence _ of (default)	Checks that the text to check appears before the specific occurrence of the text in this box. To ignore the text that appears before the text to check, clear this check box.
Constant (default)	Displays the text that appears before the text to check.
Parameter	Sets the text string as a parameter. For more information, see Chapter 7, “Parameterizing Tests.”

Checking Objects

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

Creating an Object Checkpoint

You can add an object checkpoint while recording or afterward.





To add an object checkpoint while recording:



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

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:

icon	Object	Class
	Check box	WebCheckBox
	Edit box	WebEdit
	Image	Image
	Radio button	WebRadioGroup

- 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 72.
- 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.

Click a step in your test where you want to add a checkpoint. The ActiveScreen displays the Web page corresponding to the highlighted step.

- 2** Right-click an object on the ActiveScreen and choose **Insert Checkpoint**.
- 3** The Object Selection - Checkpoint Properties dialog box opens.
- 4** Select the item you want to check from the displayed object tree. The tree item name depends on the object's class, for example:

icon	Object	Class
	Check box	WebCheckBox
	Edit box	WebEdit
	Image	Image
	Radio button	WebRadioGroup

- 5** Click **OK**. The Checkpoint Properties dialog box opens.
- 6** Specify the settings for the checkpoint. For more information, see “Understanding the Checkpoint Properties Dialog Box,” on page 72.
- 7** Click **OK** to close the dialog box.

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

Understanding the Checkpoint Properties Dialog Box

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

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

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

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

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

Edit value:

Constant: on

Parameter: roundtrip_value

In Global Data Table In Local Data Table

Use data table formula (advanced)

Regular expression

OK Cancel Help



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, the Virtual User Recorder 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.
Value	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 74.

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 7, "Parameterizing Tests."
In Global Data Table	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 12, "Working with Actions."
In Local Data Table	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 12, "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. The Virtual User Recorder 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 13, "Working with Data Tables."
Regular expression	Sets the property value as a regular expression. For more information, see Chapter 11, "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 Checkpoint Properties Dialog Box,” on page 72.

- 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 Checkpoint Properties Dialog Box,” on page 72.
- 6 Click **OK** to close the dialog box.

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

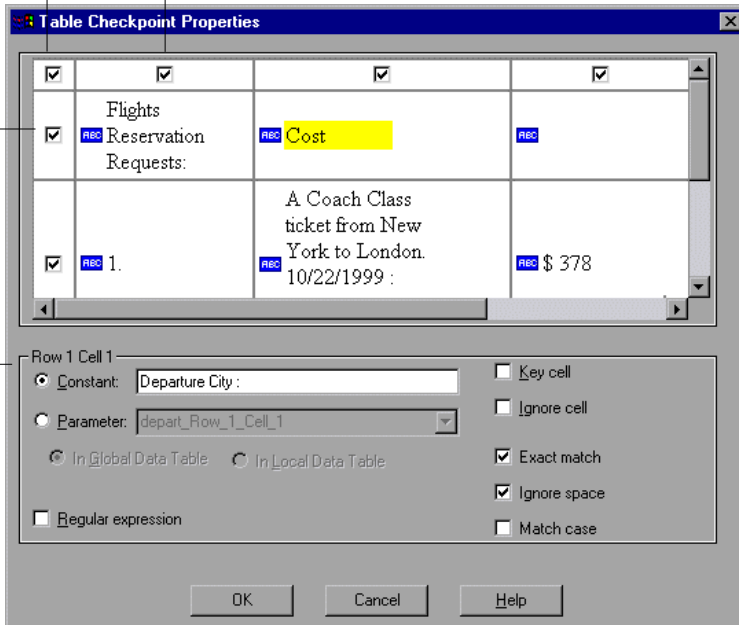
Understanding the Table Checkpoint Properties Dialog Box

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 Select/Clear
 an entire an entire
 table 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. The Virtual User Recorder will only check cells for which the corresponding row and column check boxes are selected.

To select a single cell in the table:

- 1 Clear the checkbox in the upper left hand corner of the dialog box. All the checkboxes are cleared.
- 2 Click inside the cell you want to select. The cell is highlighted.
- 3 Clear the Ignore cell checkbox. The cell is selected

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 7, “Parameterizing Tests.”
In Global Data Table	Adds a parameter to the Global tab in the Data pane. For more information, see Chapter 12, “Working with Actions.”
In Local Data Table	Adds a parameter to the Action tab in the Data pane. For more information, see Chapter 12, “Working with Actions.”
Regular expression	Sets the value of the cell as a regular expression. For more information, see Chapter 11, “Using Regular Expressions.”
Key cell	Instructs the Virtual User Recorder to search for this row according to the contents of this cell.

Option	Description
Ignore cell	Instructs the Virtual User Recorder not to check the contents of this cell.
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.

7

Parameterizing Tests

Astra LoadTest 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 the Virtual User Recorder 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 entire test or selected actions several times. In each repetition, or *iteration*, Astra LoadTest 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.

	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 LoadTest submits for all the parameters during a single iteration of the test. When you run your test, Astra LoadTest 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 : Departure Date :

Arrival City : Return Date :

No. of Passengers : Roundtrip ticket

Seating Preference Type of Seat

Aisle First
 Window Business
 None Coach

The site returns the available flights for the requested itinerary.



search results
FLIGHTS

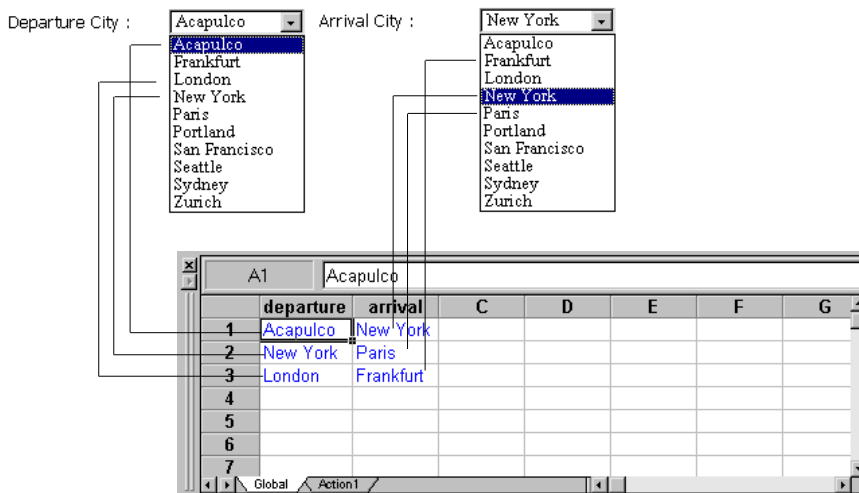
Flight departing from Acapulco to New York on
04/05/2000

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

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 LoadTest 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 10, “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 232.

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 Chapter 23, “Setting Testing Options for a Single Test.”

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 Chapter 12, “Working with Actions.”

Parameterizing Steps

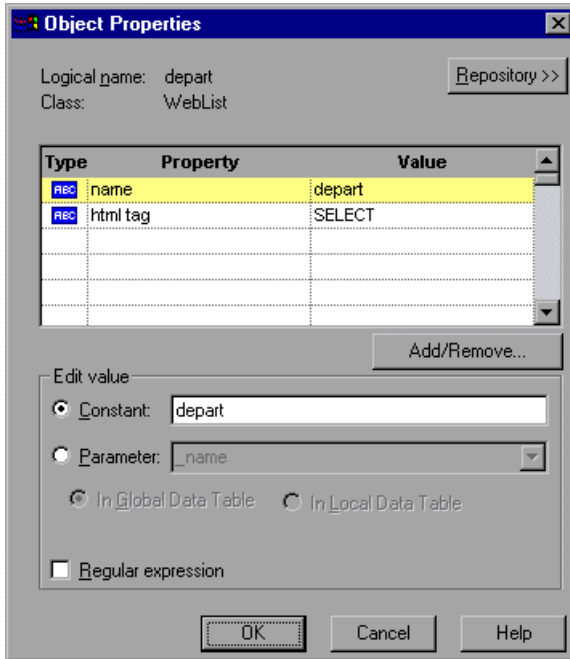
You can parameterize a step while recording your test or afterward. You parameterize a step in your test tree. A step is made up of an *object* that you navigate in your Web page, and/or a *method* by which you navigate the step. When you parameterize a step, you are actually parameterizing either the object or the method. For an example of a parameterized step, see “Example of a Parameterized Test”.

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 radio buttons. You may want to test how your site responds when different radio buttons are selected. Rather than record a separate test for clicking each radio button, you can parameterize your test so that during each iteration of the test run, Astra LoadTest clicks a different radio button.

To parameterize the object in a step:



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



The Object tab displays information about the object in the step and a link to the Object Repository:

Information	Description
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.
Repository	A link to the Object Repository. For more information about the Object Repository, see Chapter 3, "Creating Tests."

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 .

- 2** 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 about actions, see Chapter 12, “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 11, “Using Regular Expressions.”

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

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

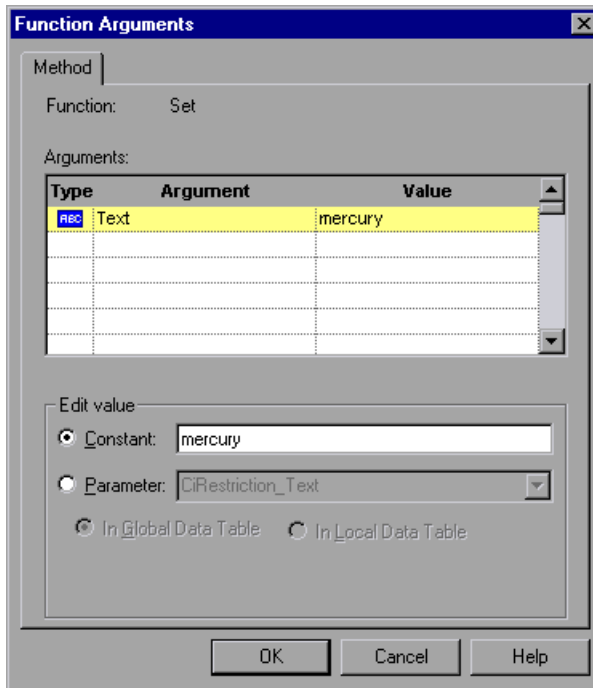
Note: You can specify additional data values for the parameter by entering them directly into the table in the Data pane. For more information, see Chapter 13, "Working with Data Tables."

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 LoadTest 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 about actions, see Chapter 12, “Working with Actions.”
- 6** Click **Close** to save the parameter and close the dialog box.

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 13, “Working with Data Tables.”

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 LoadTest checks the flight information for a different destination. For an example of a parameterized checkpoint, see “Example of a Parameterized Test” on page 91.

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

- 1 Right-click a checkpoint in the test tree and choose **Checkpoint Properties**.
 - For a page checkpoint, the Page Checkpoint Properties dialog box opens.
 - For a text checkpoint, the Text Checkpoint Properties dialog box opens.
 - For an object checkpoint, the Checkpoint Properties dialog box opens.
 - For a table checkpoint, the Table Checkpoint Properties dialog box opens.
- 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 13, “Working with Data Tables.”

- 4 To use a regular expression with the parameter, select the **Regular expression** check box. For additional information, see Chapter 11, “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.

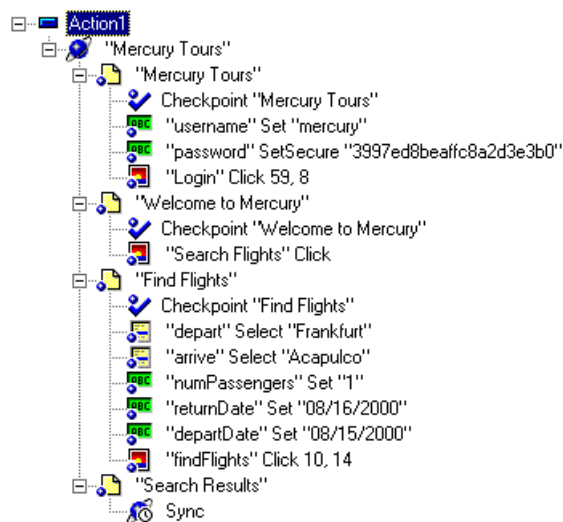
Note: You can specify additional data values for the parameter by entering them directly into the table in the Data pane. For more information, see Chapter 13, “Working with Data Tables.”

Example of a Parameterized Test

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

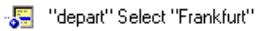
When you test your Web site, you may want to check how it performs the same operations with multiple sets of data. For example, if you are testing the sample flight Web site, “Mercury Tours,” you may want to check that the correct departure and the arrival cities are selected before you book a particular 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 LoadTest checks the flight information for a different locations.

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

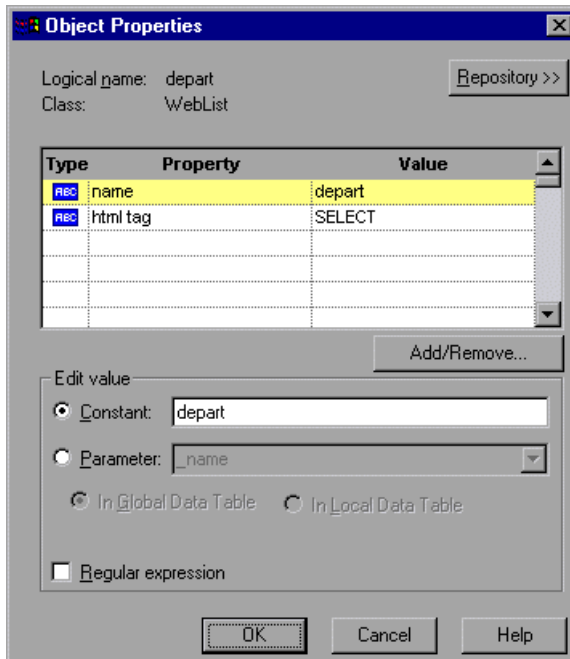


Parameterize a Step

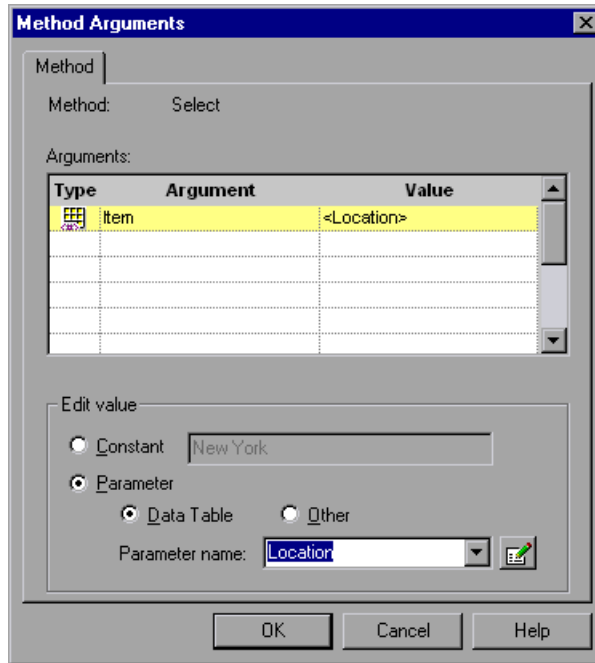
Parameterize the object and the method of the following step:

 "depart" Select "Frankfurt"

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



In the following example, parameterize the method of the above step. In the 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 83.

Parameterize a Checkpoint

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



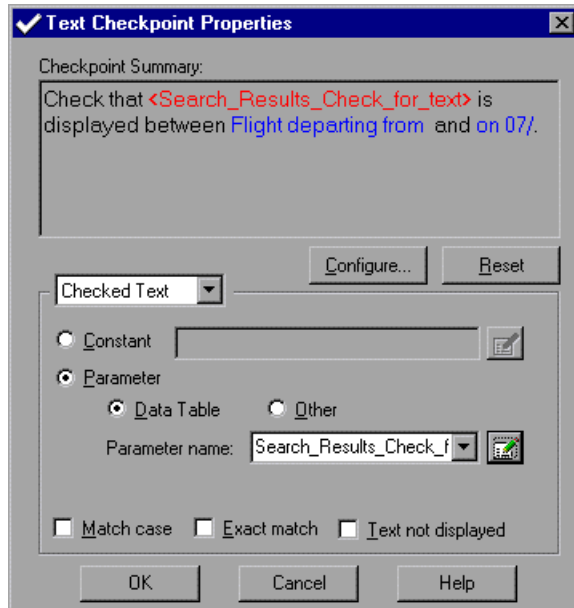
search results
FLIGHTS

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](#)

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 89.

Enter Data in the Data Table

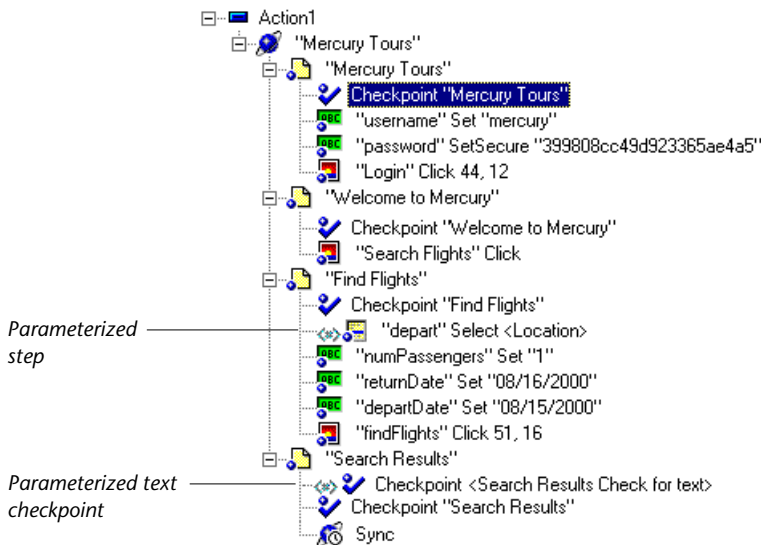
Complete the table in the Data pane. The data table may appear as follows:

	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 13, "Working with Data Tables."

Modified Test

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



8

Testing ActiveX Controls and Java Applets

Astra LoadTest supports testing on ActiveX controls in Microsoft Internet Explorer versions 5.x and higher.

This chapter describes:

- ▶ Recording and Running Tests on ActiveX Controls
- ▶ Inserting Functions with the ActiveX Wizard
- ▶ Using Scripting Functions with ActiveX Controls
- ▶ Testing Java Applets

About Checking ActiveX Controls

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

Using the Expert view and the ActiveX Wizard you can activate ActiveX control methods, retrieve and set the values of properties (including run-time properties), and use the values to enhance your test.

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 20, “Testing in the Expert View.”

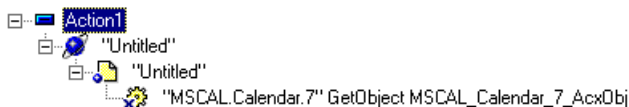
Recording and Running Tests on ActiveX Controls

The Virtual User Recorder records ActiveX controls as it does any other object. Additionally, the Virtual User Recorder identifies the functions associated with the ActiveX object. After recording, the ActiveX Wizard provides you access to these functions to enhance your test

Note: If an ActiveX object does not have an IDispatch interface and causes a navigation to take place, only the navigation will be recorded.

After running your test, the Virtual User Recorder displays the details of the steps in the Test Results window. For more information on running tests, see Chapter 15, "Running Tests in Stand-Alone Mode." For more information on viewing test results, see Chapter 16, "Analyzing Test Results in Stand-Alone Mode."

While recording a test, the Virtual User Recorder retrieves pages containing ActiveX controls. When you operate an ActiveX control, the Virtual User Recorder 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:



The Virtual User Recorder records this step in the Expert View as:

```
Browser("Untitled").Page("Untitled").WebActiveX("MSCAL.Calendar.7").GetObject MSCAL_Calendar_7_AcxObj
```

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

```
...WebActiveX ( ActiveX_control ).GetObject ActiveX_control_AcxObj
```

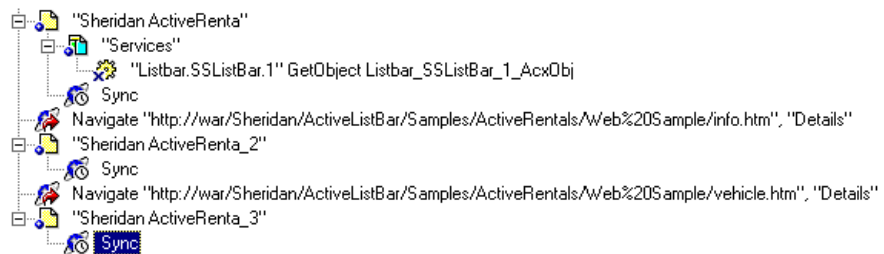
Where:

WebActiveX is the name of the test object.

GetObject is the method

ActiveX_control_AcxObj is a pointer to the real ActiveX object

The above example is a recording of a simple click on an ActiveX object. There are times however, when performing an action on an ActiveX object causes a navigation to take place. When this occurs, the navigation is recorded as a unique navigation entry. For example, if you access an online car rental site where various forms are retrieved by clicking on different areas of the ActiveX object, the Tree View may appear as follows:



The first time an operation is performed on the ActiveX object, the object is entered in the script. Additional operations are recorded beneath the ActiveX entry, but do not cause additional entries of the object itself.

In the example above, you can see that there is a single entry for the ActiveX object. Beneath it are two Navigate entries. There can be any number of Navigations recorded for a single ActiveX object.

Note: If an ActiveX object contains another ActiveX object, then the Virtual User Recorder can run and record navigations on this internal control as well. The WebActiveX test object will only be recorded for the ActiveX container, while the navigation will be recorded for both..

You run tests on ActiveX controls just as you would with any other Virtual User Recorder 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.

Note: If a test is running on one or more remote machines, make sure that ActiveX and Macromedia plug-ins are installed and updated on each machine.

For more information about running tests, see Chapter 15, “Running Tests in Stand-Alone Mode.”

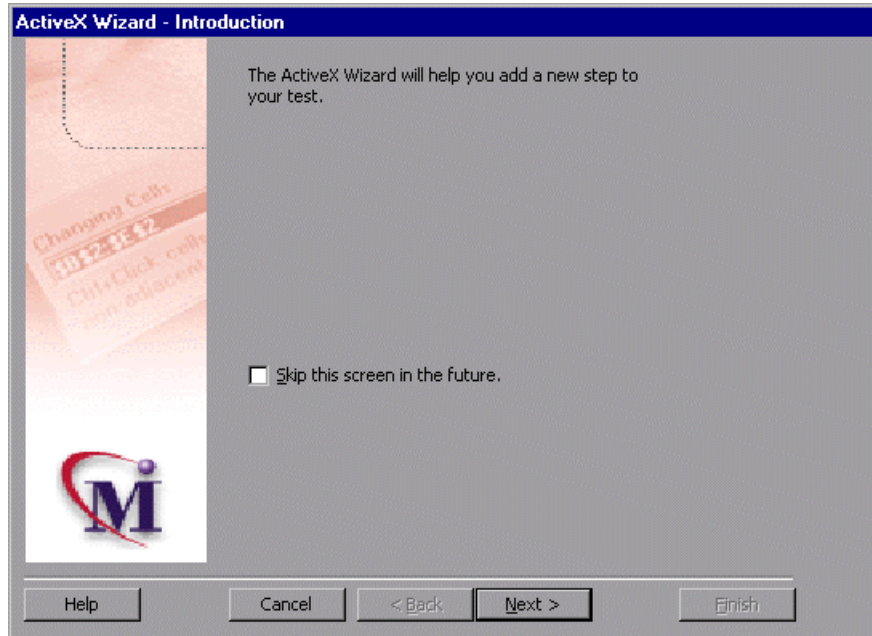
For more information about test results, see Chapter 16, “Analyzing Test Results in Stand-Alone Mode.”

Inserting Functions with the ActiveX Wizard

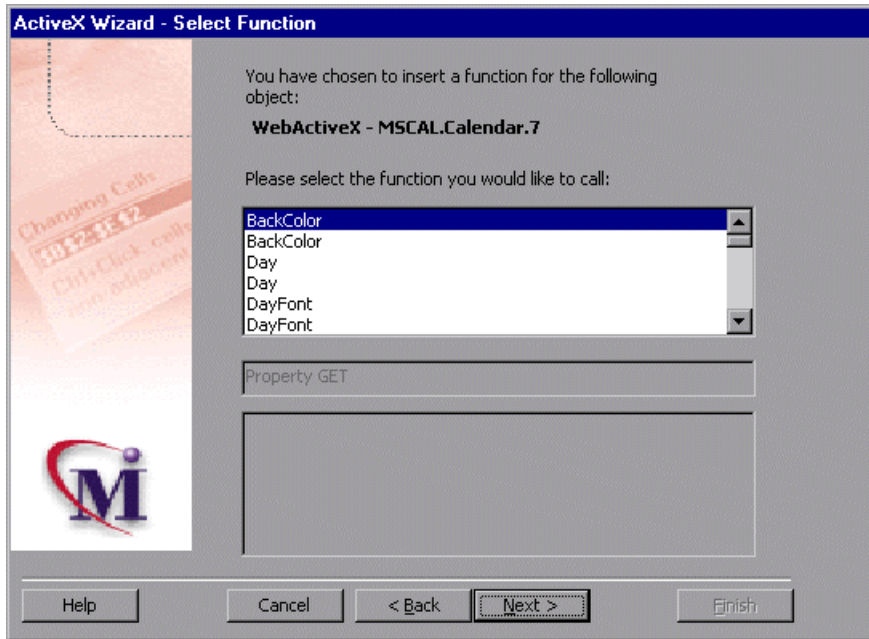
When you operate an ActiveX control, the Virtual User Recorder records the functions of the ActiveX object. The ActiveX Wizard provides a list of these functions, and allow you to easily enhance your script by inserting these functions into your test. The functions can perform operations on ActiveX objects or retrieve information from your site.

To insert a function using the ActiveX Wizard:

- 1** Select an ActiveX object in the tree view.
- 2** Choose **Insert > Step > Using the ActiveX Wizard**. The ActiveX Wizard opens to the introductory screen.



3 Click **Next**. The ActiveX Wizard - Select Function screen opens.



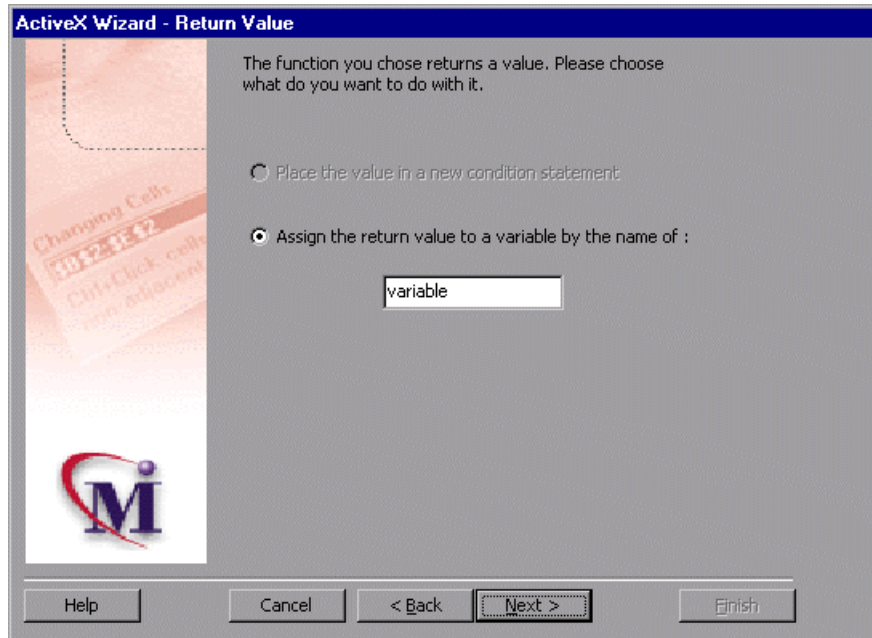
When you select a function, the function type appears in the center box. The function type can be a method or property of the ActiveX object. A property can be a GET, PUT or PUTREF.

In some cases, when an ActiveX object is created, a Help is associated with its methods. In such a case the Help that corresponds to the selected method will appear in the bottom box.

Both the function type and Help can help you understand the way a given function operates.

Select a function and click **Next**.

- 4 If the function you chose returns a value, the ActiveX Wizard - Return Value screen opens. If the function you chose has an Argument, the ActiveX Wizard - Function Arguments screen opens. Otherwise, the ActiveX - Finished screen will appear.



Depending on the function, either a Get or Put statement will be inserted into your script.

When a script containing a Get statement is run, the Controller gets the value of the property and assigns it to a variable.

When a script containing a Put statement is run, the Controller puts the value of the variable into the value of the property.

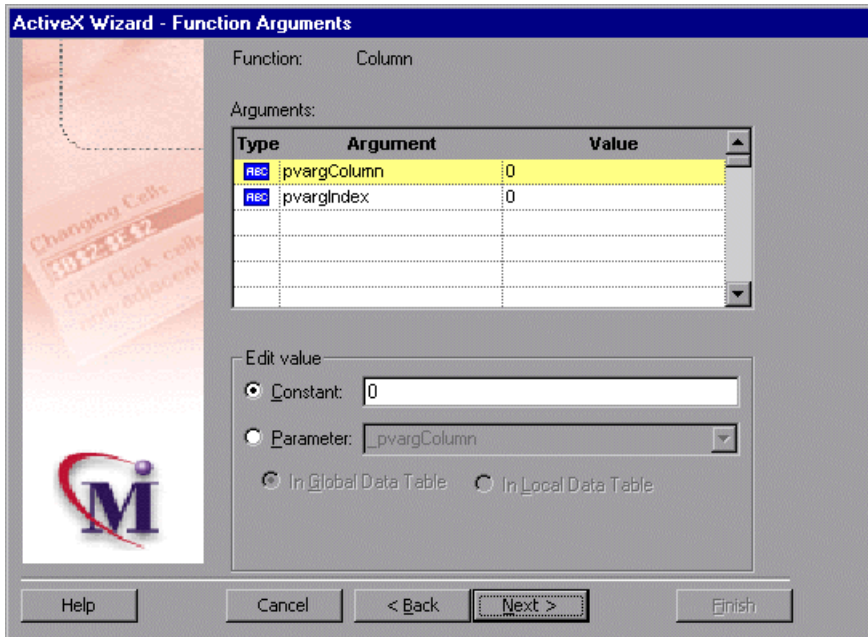
Get Property, Put Property, PutRef Property can be specified for a function.

The following options are available:



Option	Description
Place the value in a new condition statement	Inserts the return value of the function into a conditional statement. This option is only enabled when the return value is a Boolean operator.
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 proceed.

- 5 If the function you chose has function arguments, the ActiveX Wizard - Function Arguments screen opens. Otherwise, the ActiveX Wizard - Finished screen will appear.



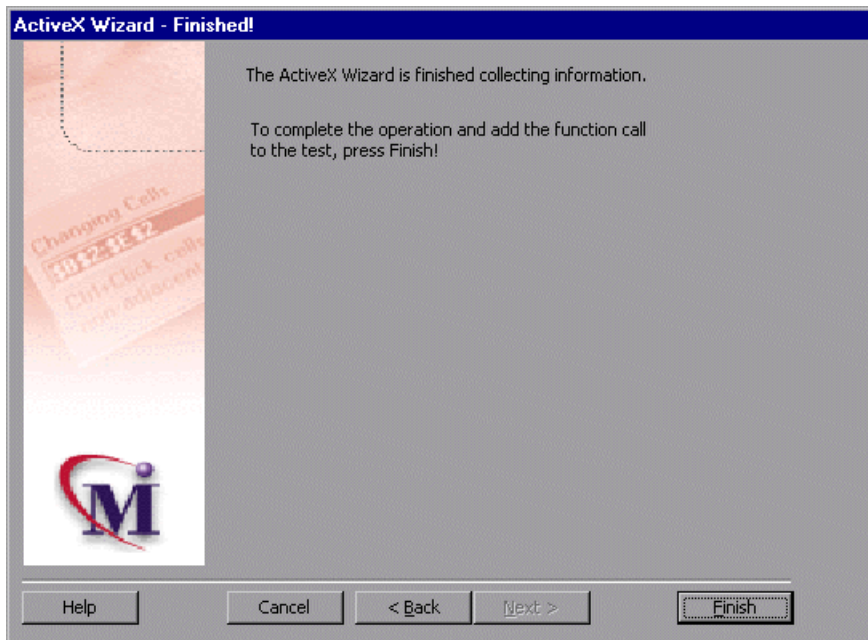
The screen displays the arguments you can parameterize, in a pane listing the arguments, their values, and their types:

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.

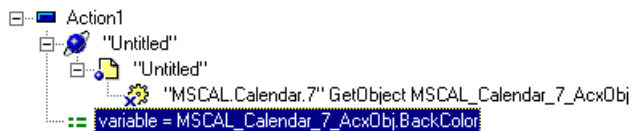
Option	Description
Constant (default)	Sets the argument value as a constant.
Parameter	Sets the argument value as a parameter. For more information, see Chapter 7, "Parameterizing Tests."

Click **Next** to continue. The ActiveX Wizard - Finished screen opens.



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

The Tree View of your test may appear as follows:



You can now enhance your test through programming the expert view. For more information on programming, see Chapter 19, "Enhancing Your Tests with Programming."

Using Scripting Functions with ActiveX Controls

The Virtual User Recorder provides several scripting functions that you can use with 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 20, “Testing in the Expert View.”

The functions are described below:

Function	Description
Exist	Checks that an object exists.
GetProperty	Returns the value of a specified property for an object.
GetObject	Returns a pointer of the object.
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 LoadTest Function Reference*.

Testing Java Applets

When recording a test, the Virtual User Recorder will only record Java applets when the applet is accessed and navigation takes place.

9

Testing Load

Today's Web applications are accessed by multiple application clients over complex architectures. With Astra LoadTest, you can emulate the load of real users interacting with your application and measure system performance.

The Virtual User Recorder enables you to customize your test to accurately measure the performance of your Web application under load.

This chapter describes:

- ▶ Inserting Transactions
- ▶ Inserting Rendezvous Points
- ▶ Setting Run-Time Options
- ▶ Run-Time Settings
- ▶ Sending Messages to Output

Inserting Transactions

To measure the performance of the server, you define *transactions*. A transaction represents a step or a set of steps that you are interested in measuring. You define transactions within your Vuser script by enclosing the appropriate sections of the script with *start* and *end* transaction statements. For example, you can define a transaction that measures the time it takes for the server to process a request to reserve a seat on a flight and for the confirmation to be displayed at the application client's terminal.

Automatic Transactions

When you record a test, Astra LoadTest automatically marks each page you browse as a transaction. This means that when you run a scenario, each page in your test tree is recognized as a transaction to be measured.

The automatic transactions create a great deal of general analysis information. As you refine your test, you may want to remove the automatic transactions and insert transactions that will measure the performance of specific business processes. For more information on enabling or disabling Automatic Transactions see, "General Settings" on page 116.

Manual Transactions

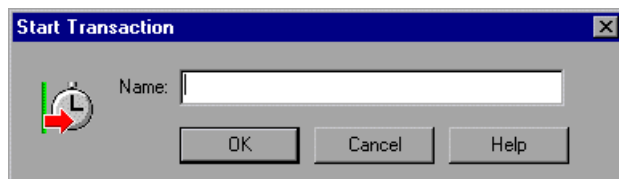
You manually insert a transaction to mark a group of steps that make up the business process that you want to measure. Manual transactions consist of a **Start Transaction** and an **End Transaction**.

A manual transaction can be inserted anywhere in your test. During replay, the Start Transaction signals the beginning of the time measurement. The time measurement continues until the End Transaction is encountered. During the replay of the test, Astra LoadTest breaks down the total time measurement into sections, supplying you with the actual time it took to download a page, or pages, and the time it took to do various actions during the recording.

There is no limit to the number of transactions which can be added to a test. You can also insert a manual transaction within a manual transaction.

To insert a transaction:

- 1 In the test tree, click the step where you want your transaction timing to begin. The page opens in the ActiveScreen.
- 2 Click the **Start Transaction** button. The Start Transaction dialog box opens.



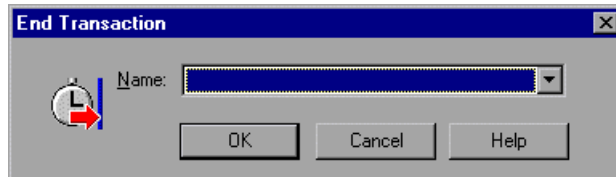
- 3 Enter a meaningful name in the **Name** box. Click **OK**.

The **Start Transaction** icon is added to the test tree above the highlighted step.

- 4 In the test tree, click the step where you want the transaction timing to end. The page opens in the ActiveScreen.

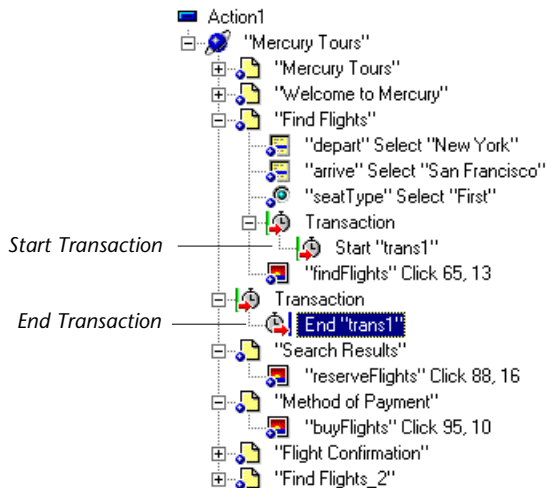


- 5 Click the **End Transaction** button. The End Transaction dialog box opens.



- 6 The Name box contains the transaction name you entered in the Start Transaction dialog box. Click **OK**.

The End Transaction icon is added to the test tree below the selected step. A sample test tree with transactions is shown below:



When running the above test, Astra LoadTest will measure the time beginning with the “findflights” Click 65,13 until the page has finished loading. Astra LoadTest will supply the time it took to perform the click, as well as the actual time it took to download the page.

Inserting Rendezvous Points

During the scenario run, you instruct multiple Vusers to perform tasks simultaneously by creating a rendezvous point. This ensures that:

- intense user load is emulated
- transactions are measured under the load of multiple Vusers

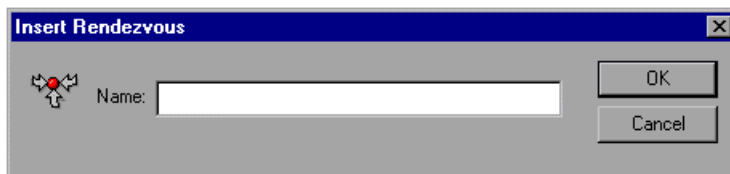
A rendezvous point is a meeting place for Vusers. To designate the meeting place, you insert rendezvous statements into your Vuser scripts. When the rendezvous statement is interpreted, the Vuser is held by the Controller until all the members of the rendezvous arrive. When all the Vusers have arrived (or a time limit is reached), they are released together and perform the next task in their Vuser scripts.

To insert a rendezvous point:

- 1** In the test tree, click the step where you want your intense user load emulated. The page opens in the ActiveScreen.

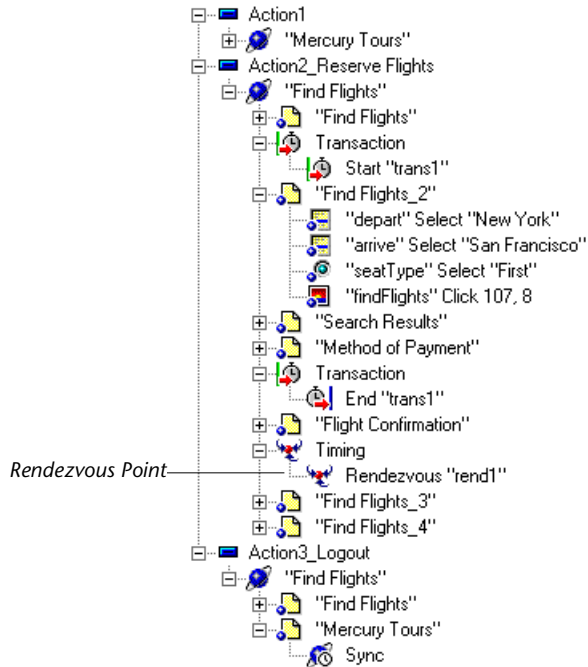


- 2** Click the **Rendezvous** button. The Rendezvous dialog box opens.



- 3** Enter a meaningful name in the **Name** box. Click **OK**.

A test tree containing a rendezvous point is shown below:



Setting Run-Time Options

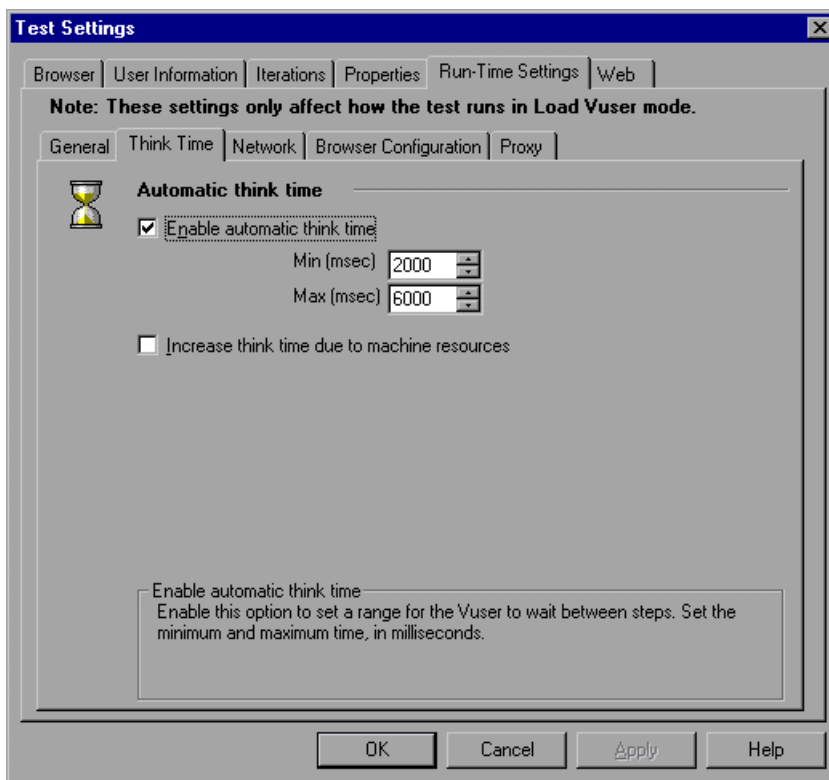
Astra LoadTest run-time options affect how your test runs in a load testing scenario. Although they are set in the Virtual User Recorder, the run-time settings are only used when load testing in the Controller. For example, you can set think time options that Astra LoadTest will use when running a test or set the output messages sent by Astra LoadTest.

Before you run your test, you can use the Run-time Settings dialog box to modify your testing options. The values you set for a given test, remain in effect for all runs of that test.

To set run-time options:

- 1 Choose **Test > Settings** and select the **Run-Time Setting** tab.

The Run-Time Settings tab is divided by subject into four tabbed pages.



- 2 Set an option, as described in “Run-Time Settings”.
- 3 When you are done, click **OK** to apply your changes and close the dialog box.

Run-Time Settings

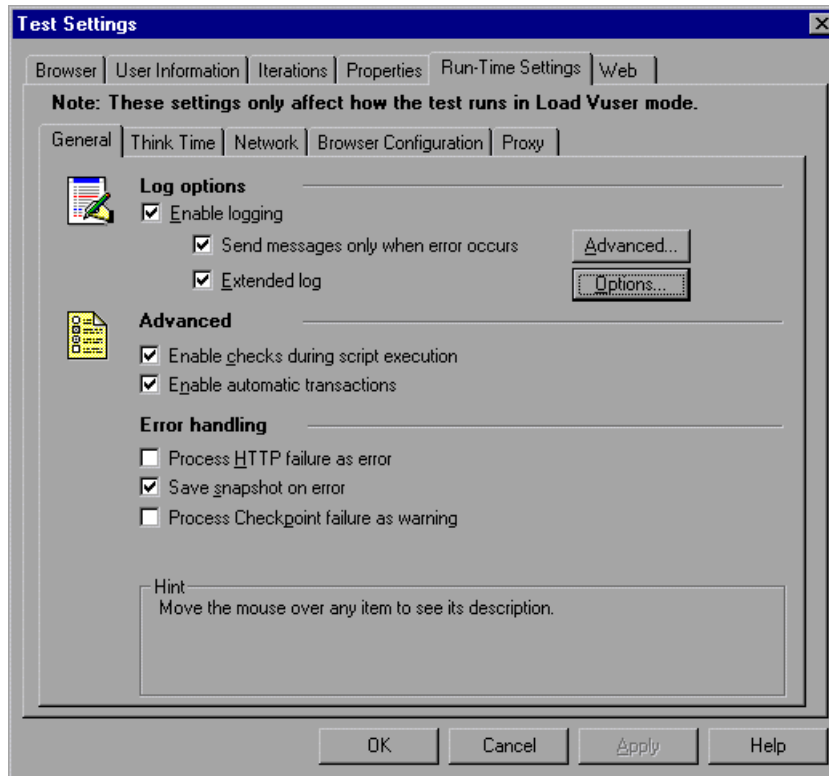
The Run-Time Settings dialog box contains the following tabbed pages:

Tab Heading	Subject
General	Options for output messages, run mode and global settings
Think Time	Options for think time
Network	Options on how to handle Web images and scripts, as well as modem speed and Cache emulation
Browser Configuration	Options for browser emulation during the running of a scenario.
Proxy	Options for supplying Proxy information.

This section lists the testing options you can set using the Run-Time Settings dialog box.

General Settings

The Log options indicate whether, and what type of, output messages Astra LoadTest should send to the output file.



The Log options include the following:

Option	Description
Enable logging	Instructs Astra LoadTest to send a subset of functions and messages sent during script execution to an output log.
Send messages only when an error occurs	Instructs Astra LoadTest to only send output messages to the log when an error occurs.
Extended log	Instructs Astra LoadTest to send detailed output messages to the output file. This information is useful when performing an Advanced Trace.

Note: The use of the log option results in a higher usage of resources, therefore, you should disable the log options to improve scalability.

The Advanced options enable Astra LoadTest to perform checks or transactions during a test run.

The Advanced options include the following:

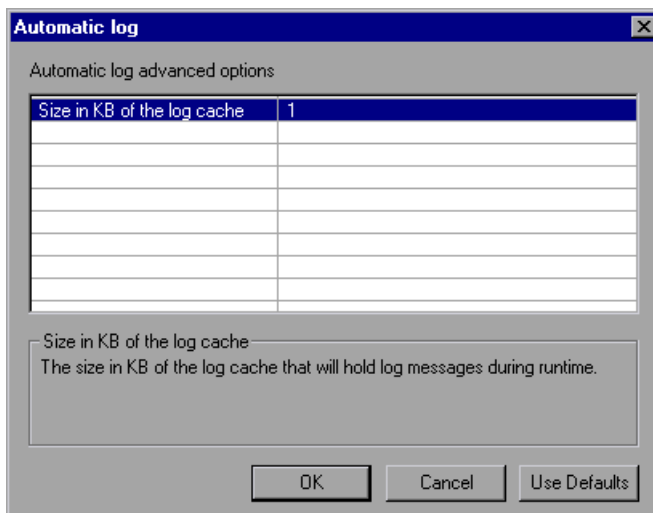
Option	Description
Enable checks during script execution	Instructs Astra LoadTest to perform checks during script execution.
Enable automatic transactions	Instructs Astra LoadTest to measure each browsed page as a transaction.

The Error handling instructs Astra LoadTest to process errors in a specific manner.

The Error Handling options include the following:

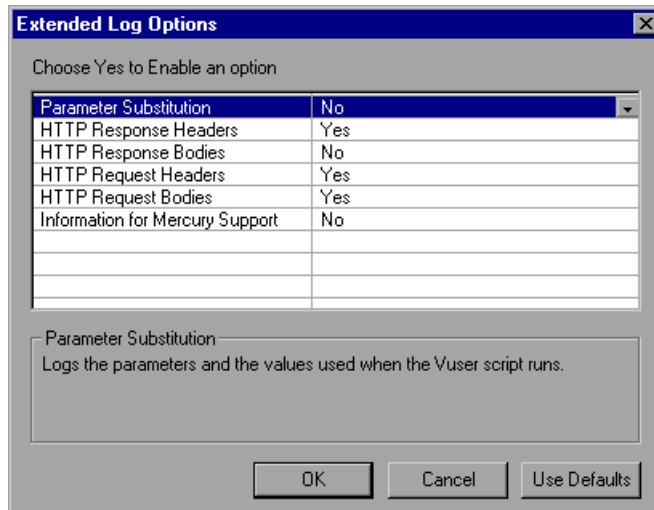
Option	Description
Process HTTP failure as error	Instructs Astra LoadTest on the handling of errors in the HTTP response header. The default value is not enabled, which will treat errors as a warning. Enabling this option causes the error to be treated as an error; failing the transaction, reporting an error to the log and failing the Vuser.
Save snapshot on error	Instructs Astra LoadTest to save an image of a Web page where an error is detected.
Process Checkpoint failure as warning	Instructs Astra LoadTest on the handling of failures in checkpoints. The default value is not enabled, which will treat failures as an error. Enabling this option causes the failure to be treated as a warning.

Selecting the Log Options Advanced button opens the Automatic log dialog box.



The Automatic log indicates the size of the log cache. When the contents of the log file exceeds the specified size, it deletes the oldest items. The default size is 1KB.

Selecting the Log Options Options button opens the Extended Log Options dialog box.



You can specify which additional information should be added to the extended log using the Extended Log Options. Select **Yes** to enable an option.

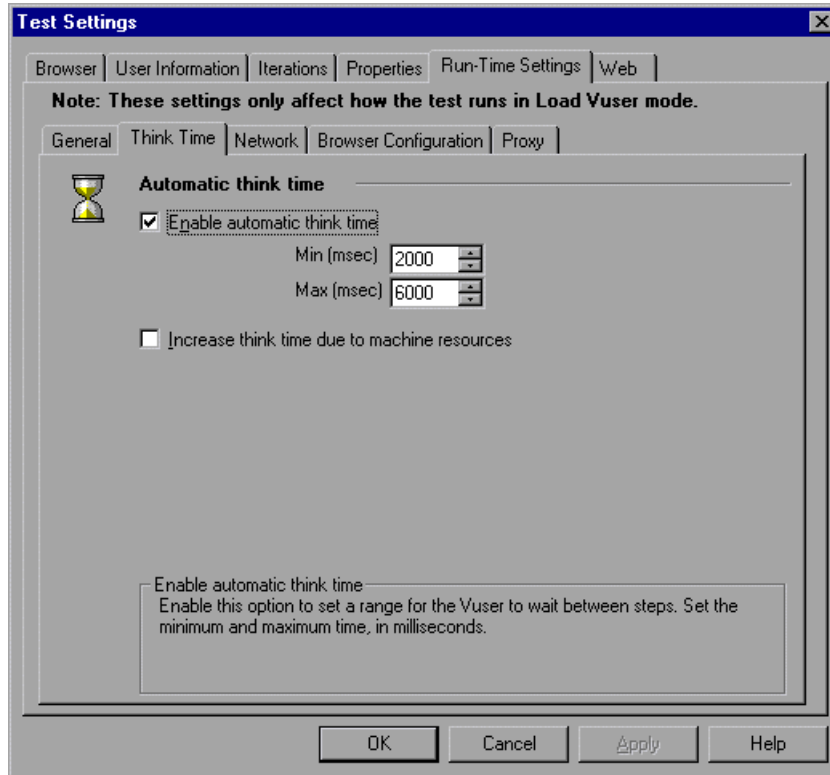
The Extended Log Options include the following:

Option	Description
Parameter Substitution	Logs the parameters and their values when the Vuser script runs.
HTTP Response Headers	Logs all HTTP Headers sent from the server to the Vuser. This option only applies to tests running with Turboload Technology.
HTTP Response Bodies	Logs all HTTP Bodies sent from the server to the Vuser. This option only applies to tests running with Turboload Technology.

Option	Description
HTTP Request Headers	Logs all HTTP Headers sent from the Vuser to the server. This option only applies to tests running with Turboload Technology.
HTTP Request Bodies	Logs all HTTP Bodiess sent from the Vuser to the server. This option only applies to tests running with Turboload Technology.
Information for Mercury Support	Information for Mercury Support. This option only applies to tests running with Turboload Technology.

Think Time Settings

The Think Time tab gives you the option to set *think time*. Think time emulates the time that a real user waits between actions.



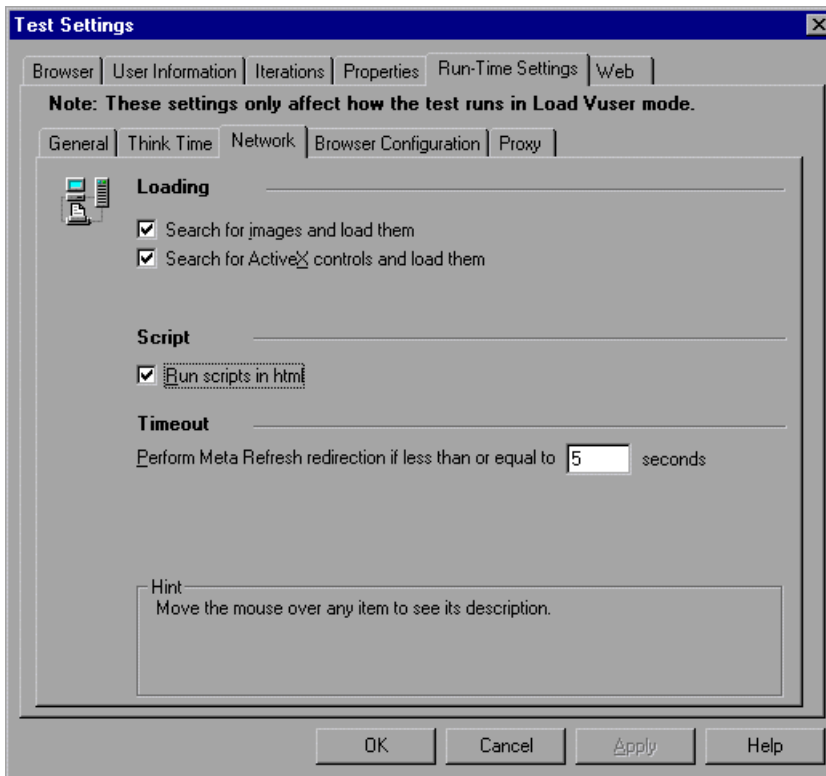
The Think Time tab includes the following options:

Option	Description
Enable automatic think time	Allows you to designate a range, in milliseconds, for the Vuser to wait between steps. Within the range, the think times are random. When think time is specified, it will be added on time value of a Transaction.
Min (msec)	Sets the minimum wait time in milliseconds.

Option	Description
Max (msec)	Sets the maximum wait time in milliseconds.
Increase think time due to machine resources	Increases the think time to compensate for slower processing times due to hardware limitations.

Network Settings

The Network tab specifies how the Vuser handles Web images, ActiveX controls, and scripts.

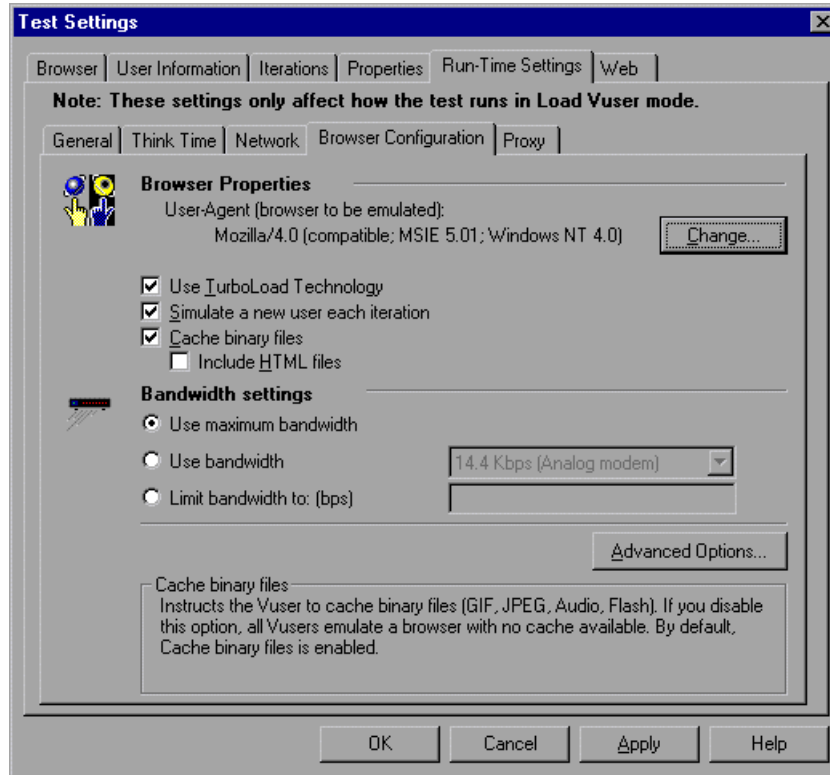


The Network tab includes the following options:

Option	Description
Search for images and load them	Instructs a Vuser to load the graphics associated with a Web page when the Vuser accesses the page during script execution.
Search for ActiveX controls and load them	Instructs a Vuser to load the ActiveX controls associated with a Web page when the Vuser accesses the page during script execution.
Run Scripts in HTML	Instructs a Vuser to run scripts associated with a Web page when the Vuser accesses the page during script execution.
Perform Meta Refresh redirection if less than or equal to _ seconds	Instructs Astra LoadTest to perform a redirection if the Meta Refresh time limit instruction (as found in the Source) is below the specified time parameter (in seconds).

Browser Configuration Settings

The Browser Configuration tab enables Astra LoadTest to emulate specific browsers, and their properties, during the playback of the scenario.

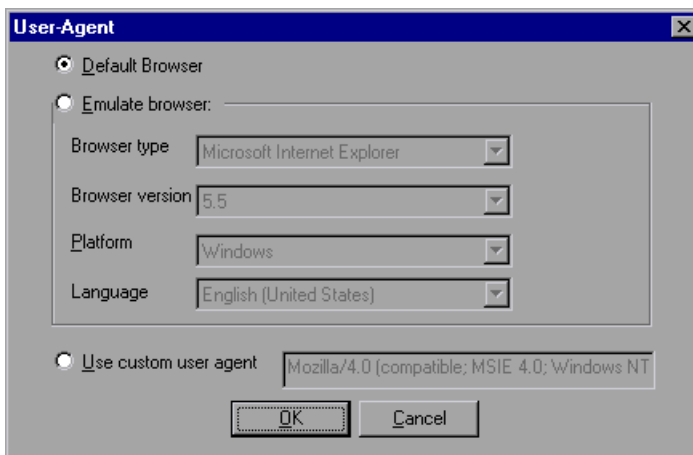


The Browser Configuration tab includes the following options:

Option	Description
Change	Opens the User-Agent dialog box where you designate the User-Agent settings.
Use TurboLoad Technology	<p>Instructs Astra LoadTest to use the TurboLoad Technology. By default, this option is enabled. When this option is disabled, the replay in the Controller is done using WinInet.</p> <p>Both TurboLoad and WinInet are ways of sending requests to, and receiving data from, the server.</p> <p>TurboLoad gives Astra LoadTest full control over socket handling, number of connections, and all the requests sent and responses received. An advantage of using TurboLoad is the ability to break the requests into their components and produce a report on them through the WebTransactionBreakdown feature.</p> <p>Disabling TurboLoad also disables the browser cache and bandwidth settings options.</p> <p>The use of TurboLoad allows the user greater control over their replay. If problems arise during the replay, switching to WinInet may solve the problem.</p>
Simulate a new user each iteration	Instructs Astra LoadTest to reset all HTTP contexts (including cookies) to their states at the beginning of the first iteration between iterations.
Cache binary files	Instructs a Vuser to cache binary files (GIF, JPEG, Audio, Flash). If you disable this option, all Vusers emulate a browser with no cache available. By default, Cache binary files is enabled.
Include HTML files	Instructs the Vuser to cache binary and txt files (HTML, .js, .css). This option is only available for use with TurboLoad. Note that using this option increases the memory footprint of each Vuser.

Option	Description
Use maximum bandwidth	Instructs Astra LoadTest to access the Web using the maximum speed that the system allows. By default, this option is enabled.
Use bandwidth	This setting allows you to indicate a specific modem speed for your Vuser to emulate. You can select any mode to emulate - from 14.4 kbps (Analog modem) to 512 kbps (DSL).
Limit bandwidth (bps)	This setting allows you to limit your bandwidth more precisely.
Advanced Options	Opens the Advanced Options dialog box

Selecting the Change button opens the User-Agent dialog box.

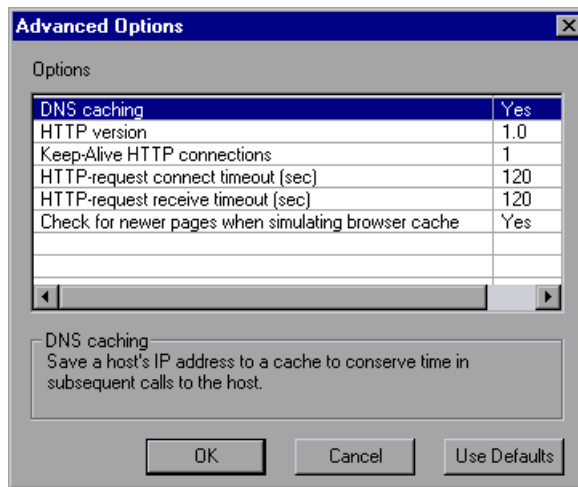


The User-Agent dialog box includes the following options:

Option	Description
Default Browser	Instructs Astra LoadTest to emulate the default browser of the host machine during playback.

Option	Description
Emulate Browser	Instructs Astra LoadTest to emulate the specified browser during playback.
Use custom user agent	Instructs Astra LoadTest to use a specified header string during playback.

Selecting the Advanced Options button opens the Advanced Options dialog box.



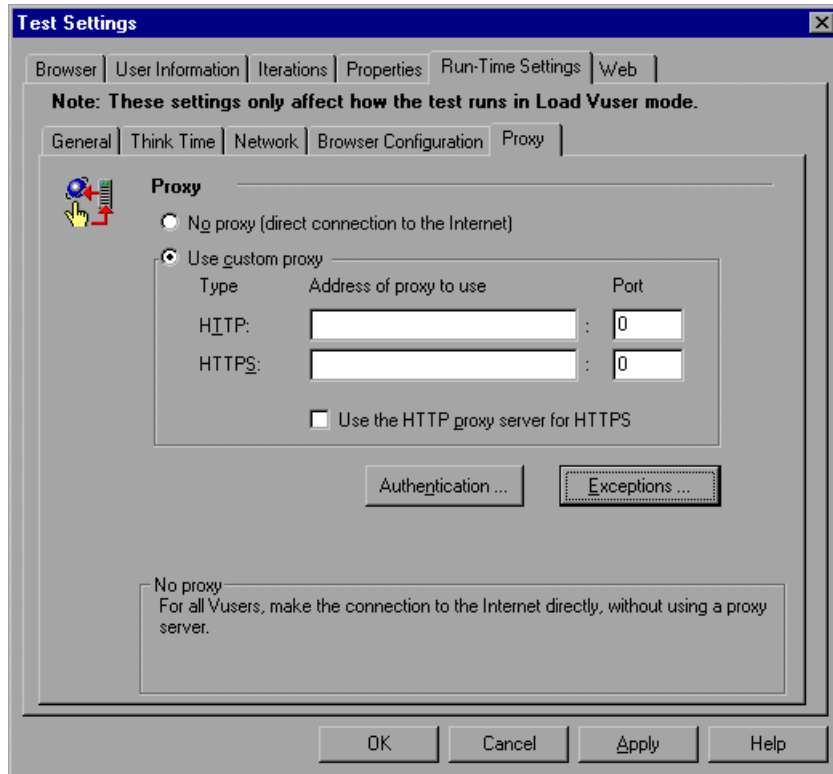
The Advanced Options dialog box includes the following options:

Option	Description
DNS caching	Instructs Astra LoadTest to save a Host's IP address to a cache. This saves time during subsequent calls to the Host.
HTTP version	Indicates the HTTP version used by your application.
Keep-Alive HTTP connections	Instructs Astra LoadTest to allow persistent HTTP connections, enabling multiple requests to be sent over the same TCP connection.

Option	Description
HTTP-request connect time-out (sec)	Instructs Astra LoadTest to allow a specified time, in seconds, to make a requested HTTP connection. If the limit is exceeded, the request will fail.
HTTP-request receive time-out (sec)	Instructs Astra LoadTest to allow a specified time, in seconds, to complete a receive operation. If the limit is exceeded, the request will fail.
Check for newer pages when simulating browser cache	Instructs Astra LoadTest to download pages that have been modified since they were cached

Proxy Settings

The Proxy tab allows you to set proxy-related settings.

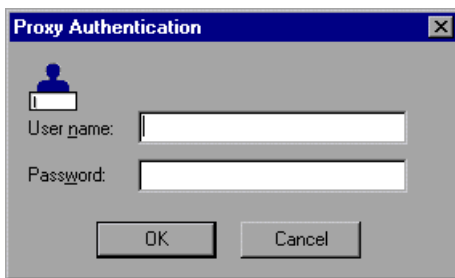


The Proxy tab includes the following options:

Option	Description
No proxy (direct connection to the Internet)	Instructs Astra LoadTest to use a direct connection to the Internet. This means the connection is made without using a proxy server.

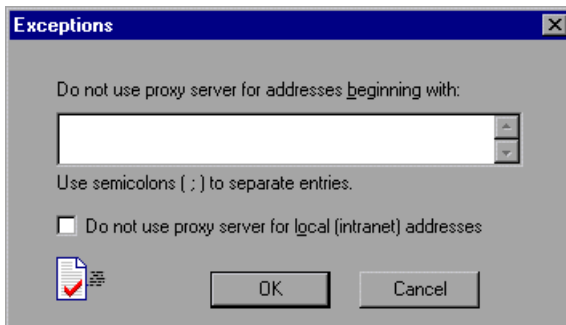
Option	Description
Use custom proxy	Instructs Astra LoadTest to use a custom proxy server. You can specify one proxy server for all HTTP sites, and another proxy server for all HTTPS (secure) sites.
Use the HTTP proxy server for HTTPS	Instructs Astra LoadTest to use the HTTP proxy server for HTTPS sites, rather than specifying a specific server for specific sites.

Selecting the Authentication button opens the Proxy Authentication dialog box.



If the proxy server requires authentication, supply the relevant password and user name.

Selecting the Exceptions button opens the Exceptions dialog box.



If you specify the use of a proxy server, you can still specify specific URLs that you want accessed in the Exception dialog box. Supply a list of these

URL's separated by semicolons. In the Exceptions dialog box, you can also specify that the Vusers should not use a proxy server for local (intranet) addresses.

Sending Messages to Output

When you run a scenario, the Controller's Output window displays information about test execution. You can include statements in a Vuser test to send error and notification messages to the Controller. The Controller displays these messages in the Output window. For example, you could insert a message that displays the current state of the Web application. You can also save these messages to a file.

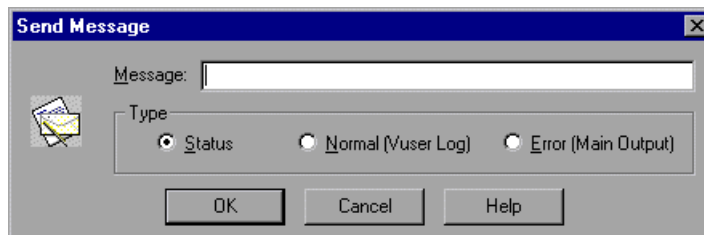
Note: Do not send messages from within a transaction. Doing so lengthens the transaction execution time and may skew the actual transaction results.

To send messages to the output window:

- 1 Click a step in your test where you want to send a message.



- 2 Click the **Send Message** button or choose **Insert > Send Message**. The Send Message dialog box opens.



- 3 Type the message into the **Message** box.
- 4 Specify the message type.

The message type includes the following options:

Option	Description
Status	Instructs Astra LoadTest to generate and print formatted output to the Controller Vuser status area.
Normal (Vuser Log)	Instructs Astra LoadTest to send the output message directly to a file.
Error (Main Output)	Instructs Astra LoadTest to send an error message to the Output window.

- 5 Click **OK** to close the dialog box. The **Send Message** icon is added above the highlighted step.

10

Creating Output Parameters

Astra LoadTest 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 LoadTest runs one iteration of the test for each set of values from your table, as discussed in Chapter 7, “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 stored as an output parameter.

When you run the test, Astra LoadTest retrieves the unique order number generated by the site for the new reservation and stores 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 LoadTest inputs the unique order number stored in the table into the order number field parameter.



You can insert 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 232.

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 LoadTest 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 Web page. 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 133.

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

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

To create a page output parameter after recording:

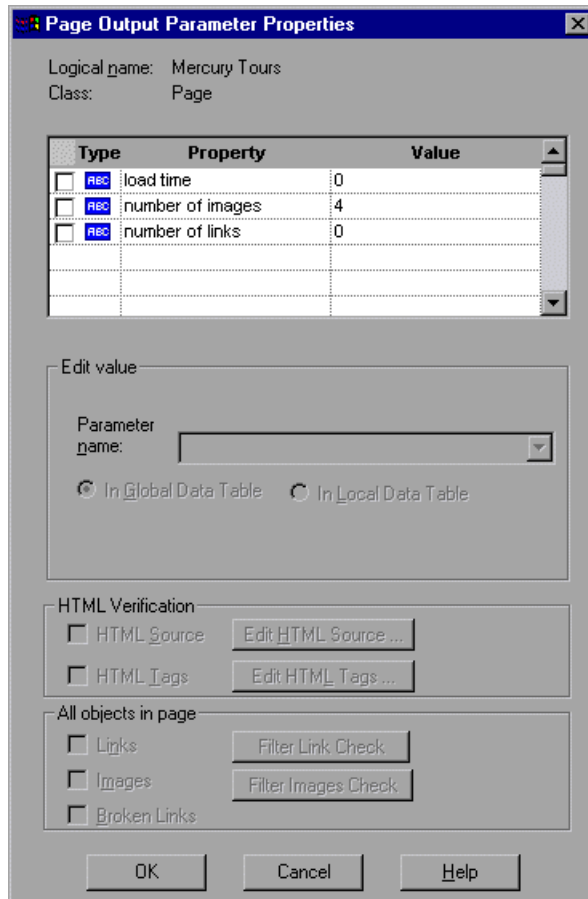


- 1** Make sure the **Display Views** button and the **ActiveScreen** tab are selected.
- 2** Click a page step in your test tree where you want to add an output parameter. The ActiveScreen displays the Web page corresponding to the highlighted step.
- 3** Right-click the page in the ActiveScreen and 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 133.
- 6** Click **OK** to close the Page Output Parameter Properties dialog box.

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.





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. This is always Page.

Choosing which Property to Parameterize

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

Pane Element	Description
Check box	To parameterize a property, select the corresponding check box.
Type	The  icon indicates that the value of the property is a constant. The  icon indicates that the value of the property is a parameter.
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 a 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. 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 <i>Working with Actions</i>
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see <i>Working with Actions</i>


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 137.
- 5** Click **OK** to close the Text Output Parameter Properties dialog box.
An output parameter tree item is added to your test tree.

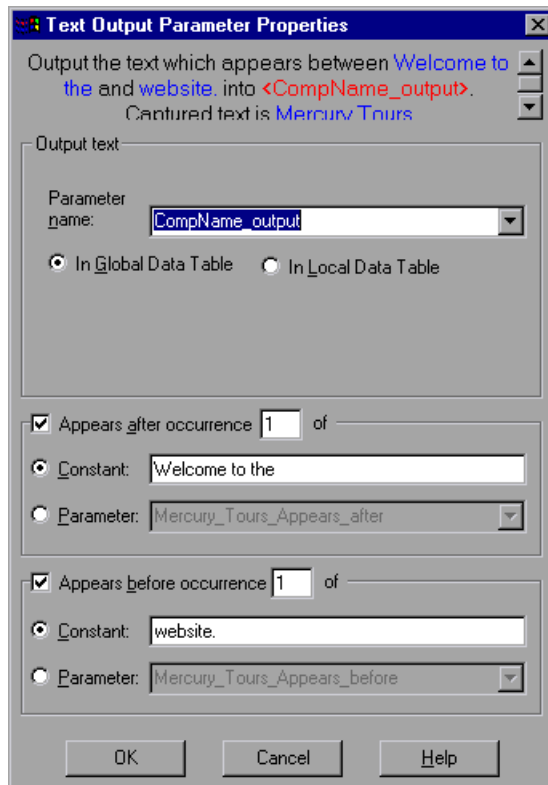
To create a text output parameter after recording:



- 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 the 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 137.
- 6** Click **OK** to close the Text Output Parameter Properties dialog box.
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. 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 Working with Actions
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see 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 before the text output parameter:

Option	Description
<p>Appears after occurrence __ of (default)</p>	<p>Outputs the text string displayed after the text specified in the Constant or Parameter box.</p> <p>If the identical text you specify is displayed more than once on the page, you can specify to which occurrence of the text you are referring.</p> <p>If you accept the default text that the Virtual User Recorder 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 non-unique text, change the occurrence number appropriately. For example, if you want to output the text string displayed after the third occurrence of the words "Mercury Tours", enter 3 in the Appears after occurrence box. To output text starting from the beginning of the page, clear this check box.</p>
<p>Constant (default)</p>	<p>Displays the text after which the text to output appears.</p> <p>Tip: If you modify the text, use a unique string whenever possible so that the occurrence number is always 1.</p>
<p>Parameter</p>	<p>Sets the text string as a parameter. To use an output parameter that you already created, select an output parameter from the list. To create a new output parameter, either use the default output parameter name or type a descriptive name for the output parameter.</p>

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
Appears before occurrence __ of (default)	<p>Outputs the text string displayed before the specified text in the Constant or Parameter box.</p> <p>the Virtual User Recorder 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 string you selected to output.</p> <p>If you accept the default text that the Virtual User Recorder recommends, the number in the dialog box will be correct. If you modify the recommended text and the text you specify is displayed in the text string to output as well as in the Appears before text, you need to modify the occurrence number accordingly. To output all text on the page that appears after the text specified in the Appears After option, clear this check box.</p>
Constant (default)	<p>Displays the text before which the text to check should appear.</p> <p>Tip: If you modify the text, use a unique string whenever possible so that the occurrence number is always 1.</p>
Parameter	<p>Sets the text string as a parameter. To use an output parameter that you already created, select an output parameter from the list. To create a new output parameter, either use the default output parameter name or type a descriptive name for the output parameter.</p>

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

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

- 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 143.
- 6** Click **OK** to close the Output Param Properties dialog box.
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 the highlighted step.

- 3** Right-click the object for which you want 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:

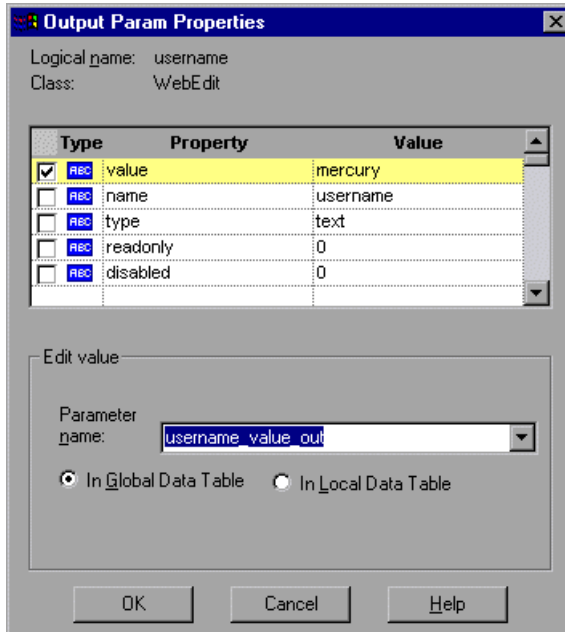
Object	Class
Check box	WebCheckBox
Edit box	WebEdit
Radio button	WebRadio
List box	WebList
Element	WebElement

- 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 143.
- 7** Click **OK** to close the Output Parameter Properties dialog box.

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 "WebRadioGroup" class indicates that the object is a radio button.

Choosing which Property to Parameterize

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

Pane Element	Description
Check box	To parameterize a property, select the corresponding check box.
Type	The  icon indicates that the value of the property is a constant. The  icon indicates that the value of the property is a parameter.
Property	The name of the property to parameterize.
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. 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 Working with Actions
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see 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 **WebTable** 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 146.

- 5 Click **OK** to close the Table Output Parameter Properties dialog box.

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 the highlighted step.

- 3 Right-click the table you want to parameterize in the ActiveScreen and choose **Output**. The Object Selection - Output Param Properties dialog box opens.



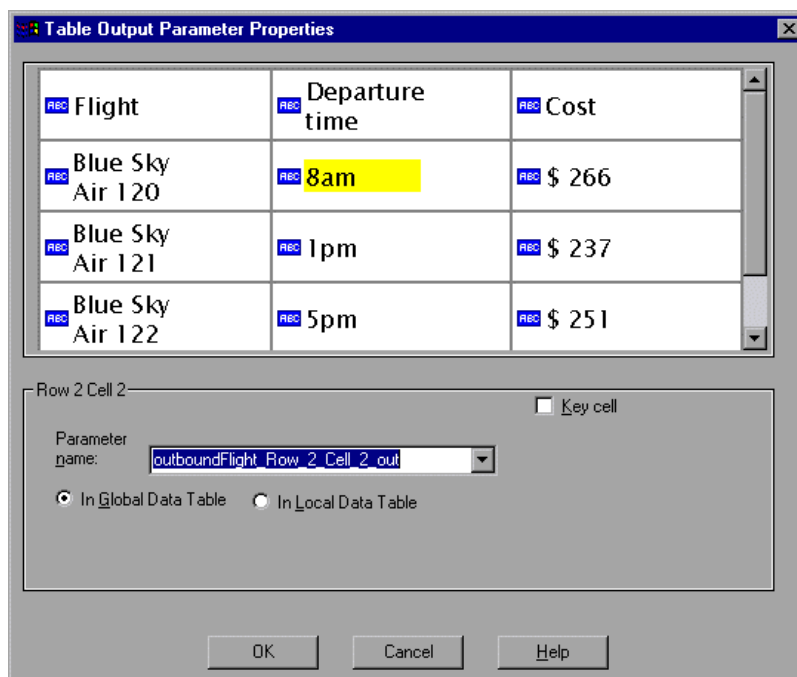
- 4 Select a **WebTable** item and click **OK**. The Table Output Parameter Properties dialog box opens.

- 5 Specify the settings for the output parameter. For more information, see “Understanding the Table Output Parameter Properties Dialog Box” on page 146.
- 6 Click **OK** to close the Table Output Parameter Properties dialog box.

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.



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, either 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 Working with Actions
In Local Data Table	Adds the output parameter name to the Action tab in the Data pane. For more information, see 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.

11

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 LoadTest 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 LoadTest searches for the literal character.

Using Regular Expressions in Steps

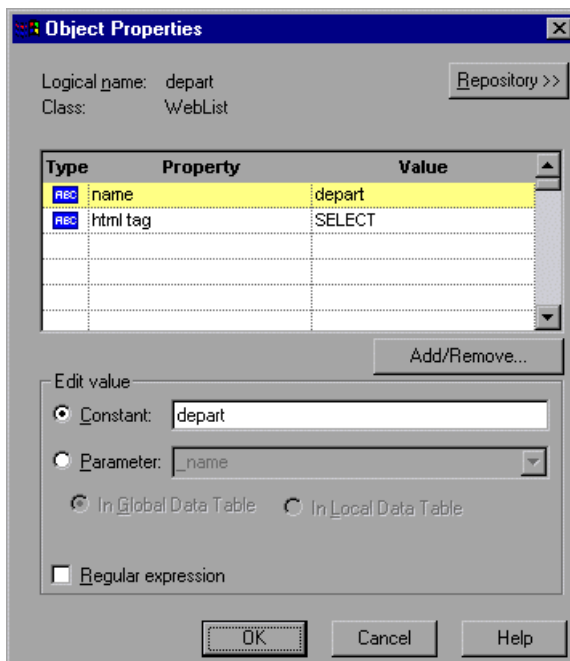
You can use regular expressions when defining or parameterizing a step in your test tree. A step is made up of an *object* that you navigate in your Web page, and/or a *method* by which you navigate the step. You can use regular expressions when defining or parameterizing the object of a step.

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 LoadTest ignores variations in the button name when clicking the button.

To define a property value as a regular expression:

- 1 Right-click a step in the test tree and choose **Object Properties**.



The Object Properties dialog box opens.



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 In the **Properties** section, click the property you want to set as a regular expression. The property is highlighted.

3 In the **Edit value** section, 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”.

- 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 7, “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 12, “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”. For information on editing the table, see Chapter 7, “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 LoadTest treats all characters in a regular expression literally, except for the 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 LoadTest treats it as literal character.

- Click **Yes** to instruct Astra LoadTest to treat a special character literally. The special character is now preceded by a backslash (\).

- ▶ Click **No** to instruct Astra LoadTest to treat all the characters literally, except for special characters.
- 6 Click **OK** to save and close the Object 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.

Using Regular Expressions in Object Checkpoints

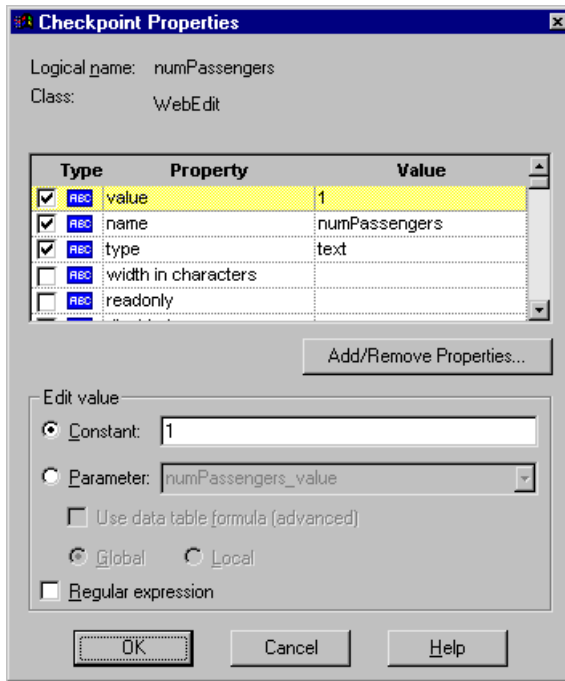
When creating an object checkpoint to verify that an object appears on your Web site, you can also set the property value of the object as a regular expression, so that an object with a varying name can be verified.

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 LoadTest 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”.

- 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 7, “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 12, “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”. For information on editing the table, see Chapter 7, “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 LoadTest 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 LoadTest treats it as literal character.

- Click **Yes** to instruct Astra LoadTest to treat a special character literally. The special character is now preceded by a backslash (\).

- ▶ Click **No** to instruct Astra LoadTest 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.

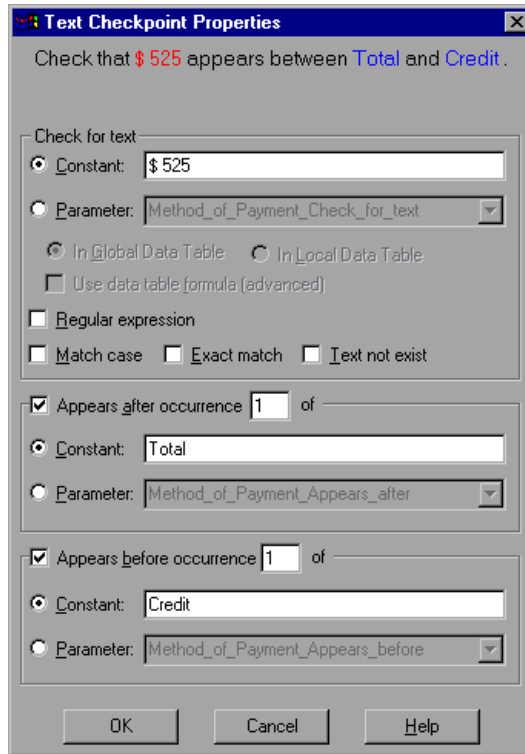
Using Regular Expressions in Text Checkpoints

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 LoadTest 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”.

- 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 7, “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 12, “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”. For information on editing the table, see Chapter 7, “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 LoadTest 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 LoadTest treats it as literal character.

- Click **Yes** to instruct Astra LoadTest to treat a special character literally. The special character is now preceded by a backslash (\).

- Click **No** to instruct Astra LoadTest 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”.
- 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 LoadTest 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 LoadTest searches for the literal character.

The following options can be used to create regular expressions:

Matching Any Single Character

A period (.) instructs Astra LoadTest 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 LoadTest 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.

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 LoadTest 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 LoadTest 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 LoadTest to match zero or one occurrences of the preceding character. For example:

```
ca?r
```

matches car and cr, but nothing else.

Matching Either the Preceding or Following Expression

A vertical line (|) instructs Astra LoadTest 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 LoadTest 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 ([: :]) instruct Astra LoadTest 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 LoadTest 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 LoadTest 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 LoadTest 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 (

12

Working with Actions

You can divide your test into actions in order to streamline the testing process of your Web sites.

This chapter describes:

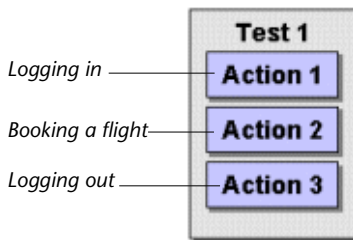
- Using Multiple Actions in a Test
- Using Global and Action Data Sheets
- Using the Action Toolbar
- Creating New Actions
- Inserting Existing Actions
- Nesting Actions
- Setting Action Properties
- Removing Actions From a Test
- Guidelines for Working with Actions

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.

Suppose you want to test several features of a flight reservation system. You plan several tests to test various business processes, but each one requires the same login and logout steps. You can create one action that contains the steps required for the login process, another for the logout steps, and other actions for the main steps in your test. Once you've created the login and logout actions, you can insert those actions into other tests.

If you created a test in which you logged in to the system, booked one flight, and then logged out of the system, your test might be structured as shown:



Actions enable you to parameterize specific components of a test and to easily re-record one action so that you don't have to re-record the entire test when part of your application changes.

Two or more tests can *call* (link to) the same action and one action can call (link to) another action. Complex tests may have many actions, and may share actions with other tests.

Using Multiple Actions in a Test

When you create a test, it includes one action. All the steps you record and all the modifications you make after recording are part of a single action.

You can divide your test into multiple actions by creating new actions or by inserting existing actions. There are two kinds of actions:

- **non-reusable action** - an action that can only be used in the test in which it was created, and only once.
- **reusable action** - an action that can be called multiple times by the test in which it was created.

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.

For every action in your test, the Virtual User Recorder creates a corresponding *action* sheet in the Data pane so that you can enter parameters that are specific to that action only. For more information on global and action data sheets, see “Using Global and Action Data Sheets”. For information on parameterizing tests, see Chapter 7, “Parameterizing Tests,” and Chapter 10, “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 16, “Analyzing Test Results in Stand-Alone Mode.”

Note: You can also run a single action, or part of an action, using the Run Action from Step option. For more information, see Chapter 15, “Running Tests in Stand-Alone Mode.”

Using Global and Action Data Sheets

When you add a parameter to your test, you can specify whether to store the data in the *global* data sheet or in the *action* data sheet. You set this in the Object Properties, Checkpoint Properties or Function Arguments dialog box for parameters or the Output Value Properties dialog box for output parameters.

- Choosing **In Global Data Table** creates columns in the **Global** sheet in the Data pane. When you run your test, the Virtual User Recorder inserts or

outputs a value from or to the current row of the global data sheet during each *global iteration*. You can use the columns in the global data sheet for output parameters or parameters in any action. This enables you to pass information from one action to another.

- Each action also 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 a parameter (column) in the corresponding action sheet in the Data pane. When there are parameters in an action's local sheet, you can set the Virtual User Recorder to run one or more iterations on that action before continuing with the current global iteration of the test. When you set your action properties to run iterations on all rows, the Virtual User Recorder inserts the next value from the action's data sheet into the corresponding action parameter during each *action iteration*, while the values of the global parameters stay constant.

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.

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.

For more information on the data table, see Chapter 13, “Working with Data Tables.”

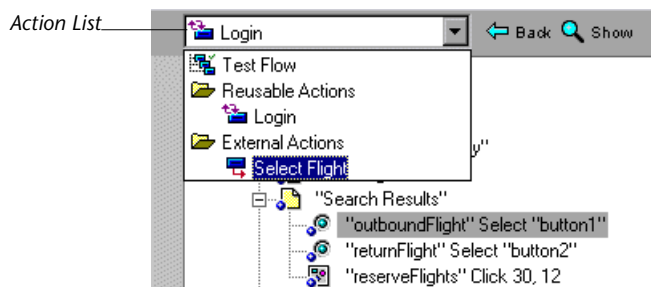
For more information about parameterization, see Chapter 7, “Parameterizing Tests.”

For more information about output parameters, see Chapter 10, “Creating Output Parameters.”

Using the Action Toolbar

By default, the Action toolbar is not displayed in the Tree View when the Virtual User Recorder opens. You can choose to view it at any time by enabling the **Display Action toolbar in Tree View** option from the General tab of the Options dialog box. For more information, see Chapter 22, “Setting the Virtual User Recorder Testing Options.”

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



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

In the test flow, reusable actions are not expandable. You can view the expanded tree of reusable actions by opening the action tree for that action. For more information about reusable actions, see “Setting the Reusability Status of an Action”.

There are three ways to switch to the action tree for a reusable 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.



If you are working in a test without reusable actions, you can hide the Action toolbar. To hide the Action toolbar, clear the **Display Action toolbar in the Tree View** option in the General Tab of the Options dialog box. For more information, see Chapter 22, “Setting the Virtual User Recorder Testing Options.”

In the Expert view, the Action toolbar is always visible, and the Expert view always displays the script for the selected action.

Creating New Actions

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

You can add the action as a top-level action, or you can add the action as a sub-action (or *nested action*) of an existing action in your test. For more information, see “Nesting Actions”.

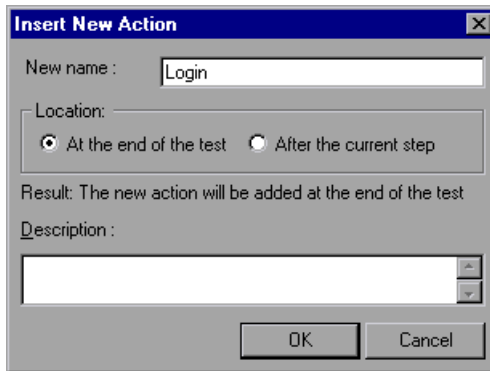
You can also split an existing action into two actions.

To create a new action in your test:

- 1** If you want to insert the action within an existing action, click the step after which you want to insert the new action.



- 2 Choose **Insert > New Action** or click the **New Action** button. The Insert New Action dialog box opens.



- 3 Type a new action name or accept the default name.
- 4 Decide where to insert the action:
 - To insert a new action at the end of your test, select **At the end of the test**.
 - To insert the new action within the action of the currently selected step, select **After the current step**.

For more information about inserting actions within actions, see “Nesting Actions”.

- 5 If you wish, add a description of the action. You can also add an action description at a later time in the Action Properties dialog box.

Tip: Descriptions of actions are displayed in the Insert Copy of Action and Insert Call to Action dialog boxes. The description makes it easier for you to select the action you want to insert. For more information, see “Adding or Editing an Action Description”.

- 6 Click **OK**. A new action is added to your test and is displayed at the bottom of the test tree or after the current step. You can move your action to another location in your test by dragging it to the desired location.



7 Make sure your new action is selected and click **Record** to continue recording. The steps you record will be added to your new action.

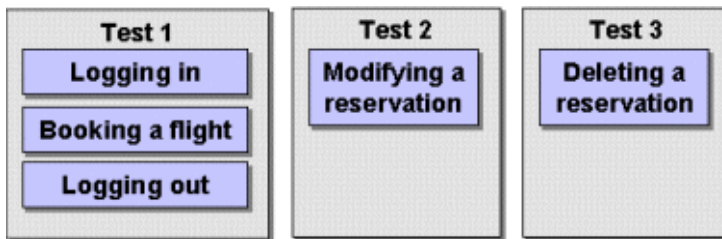
By default, all new actions are inserted as non-reusable actions. If you want to be able to call the action from within your test or from other tests, you can make it a reusable action. For more information about reusable actions, see “Setting the Reusability Status of an Action”.

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 a flight reservation system in one test. Once you are satisfied with the action you recorded and enhanced, you can insert the existing action into other tests.

You can insert an existing action by inserting a copy of the action into your test, or by inserting a call to the original action.

Suppose you wanted to record the following three tests in the Mercury Tours site: Booking a flight, Modifying a reservation, and Deleting a reservation. While planning your test you realize that for each test you need to log in and log out of the site. If you plan to use the insert existing action option for the repeated activities, then you would initially record the three tests as shown:



Once you have set up your tests, you must choose whether you want to insert a copy of the action or insert a call to it.

About Reusable Actions

You can call a reusable action multiple times within a test (from the *local test*), and you can call it from other tests. You can insert copies of non-reusable actions into your test, but you cannot insert calls to non-reusable actions.

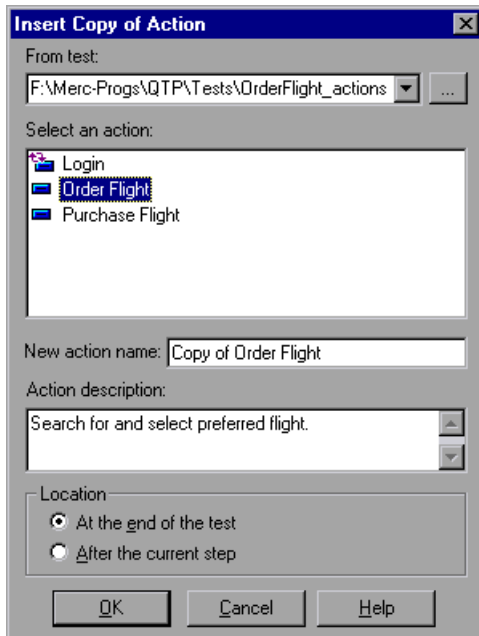
Inserting calls to reusable actions makes it easier to maintain your tests, because when an object or procedure in your application changes, it only needs to be updated one time, in the original action.

Inserting a Copy of an Action

When you insert a copy of an action into a test, the action is copied in its entirety, including checkpoints, parameterization, and the corresponding action tab in the Data pane. The action is inserted into the test as an independent, non-reusable action (even if the original action was reusable). Once the action is copied into your test, you can add to, delete from, or modify the action just as you would with any other recorded action. Any changes you make to this action after you insert it affect only this action, and changes you make to the original action do not affect the inserted action. You can insert copies of both reusable and non-reusable actions.

To insert a copy of an action:

- 1** Choose **Insert > Copy of Action** or right-click the action and select **Insert Copy of Action**. The Insert Copy of Action dialog box opens.



- 2** Use the browse button to find the test from which you want to insert the action. The action window displays all *local* actions (actions that originated) in the test you selected.
- 3** Select the action you want to insert from the list. When you highlight an action, its description, if one exists, is displayed in the Action Description box. This helps you identify the action you want to insert. For more information about action descriptions see “To add or edit an action description:” on page 181.
- 4** Type a meaningful name for the action in the **New Action Name** box.
- 5** If you wish, add or modify the action's description.

6 Decide where to insert the action:

- To insert a new action at the end of your test, select **At the end of the test**.
- To insert the new action within the action of the currently selected step, select **After the current step**.

For more information about inserting actions within actions, see “Nesting Actions”.

- 7** Click **OK**. The action is inserted into the test as an independent, non-reusable action. You can move the action to another location in the test by dragging it to the desired location in the test tree.

Inserting a Call to an Action

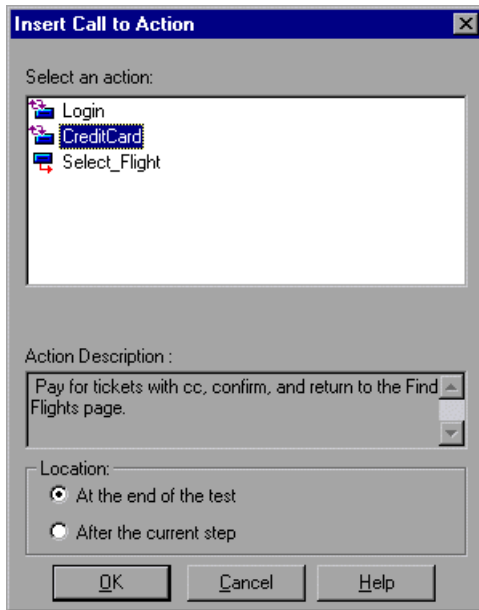
You can insert a call (link) to a reusable action that resides in your current test (local action).

To modify the action, you must open the test in which the action originated. In the original action, the modifications apply to all tests that call that action. If you chose to use the original action’s data, then changes to the original action’s data will be applied as well.

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

To insert a call to an action:

- 1 Choose **Insert > Call to Action** or right-click the action and select **Insert Call to Action**. The Insert Call to Action dialog box opens.



- 2 Select the action you want to insert from the list. When you highlight an action, its description, if one exists, is displayed in the Action Description box. This helps you identify the action you want to insert. For more information about action descriptions see “To add or edit an action description:” on page 181.
- 3 Choose where to insert the action:
 - To insert a new action at the end of your test, select **At the end of the test**.
 - To insert the new action within the action of the currently selected step, select **After the current step**.

For more information about nesting actions, see “Nesting Actions”.

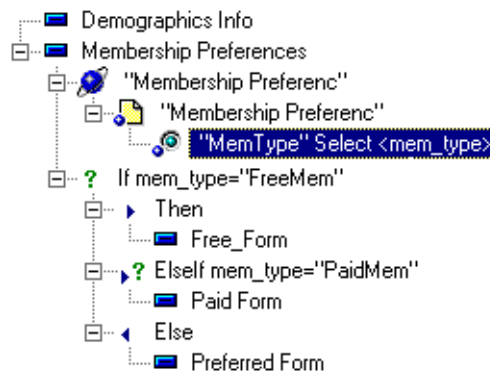
- 4 Click **OK**. The action is inserted into the test as a call to the original action. You can move the action to another location in the test by dragging it to the desired location in the test tree.

Nesting Actions

Sometimes you may want to run an action within an action. This is called *nesting*. Nesting actions can:

- help you maintain the modularity of your test
- enable you to run one action or another based on the results of a conditional statement

For example, suppose you have parameterized a step where a user selects one of three membership types as part of a registration process. When the user clicks **Continue** on the registration form, the page that opens depends on the membership type selected in the previous page. You can create one action for each type of membership. Then you can use *if* statements to determine which membership type was selected in a particular iteration of the test, and run the appropriate action for that selection. Your script might look something like this:



To nest an action within an existing action:

- 1 Highlight the step after which you would like to insert the action.
- 2 Follow the instructions for creating a new action as described on “Creating New Actions”, or follow the instructions for inserting an existing action as described on “Inserting Existing Actions”.
- 3 In the Location section of the Insert New Action, Insert Copy of Action or Insert Call to Action dialog box, select **After the current step**. The action is inserted after the current step.

Note: In the Expert View, an action called by another action is displayed within the parent action with the following syntax:

Call RunAction (*ActionName*, *Run_Setting*, *FromRow - ToRow*)

For example:

Call RunAction("Select Flight", 0, "1 - 1")

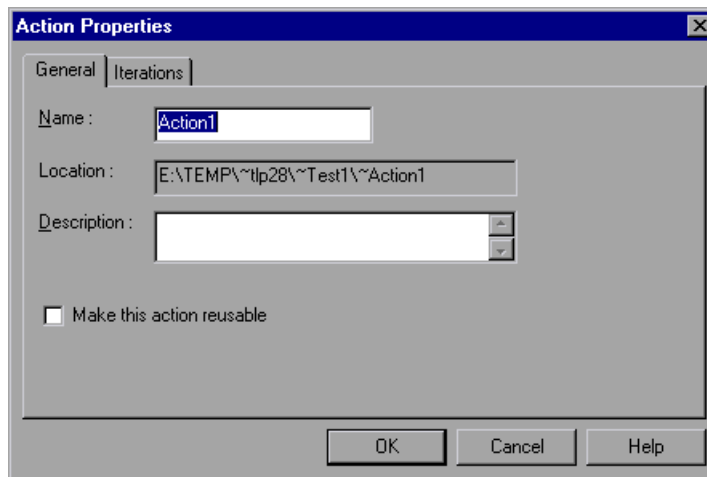
Setting Action Properties

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.

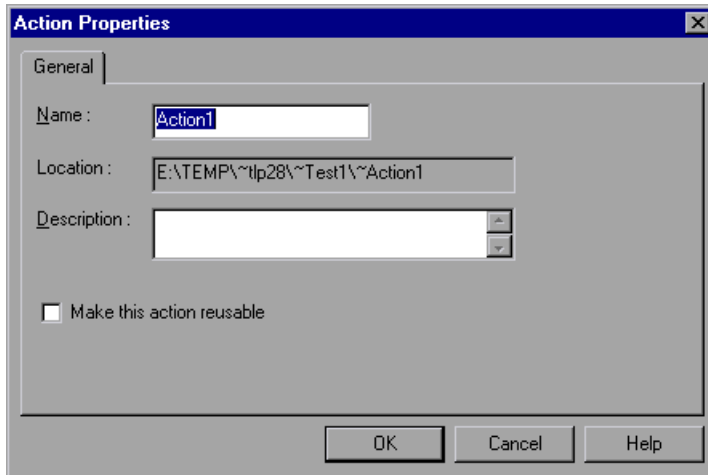
Opening the Action Properties Dialog Box

You can open the Action Properties dialog box from the Tree View with the test flow displayed, from the Tree View with an action tree 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 actions, or with **Test Flow** selected in the Action List), the Action Properties dialog box contains both the General tab and the Run tab as shown below:



When you open the Action Properties dialog box from an action tree (either from the Tree View or the Expert View with a specific action displayed in the Action List), the Action Properties dialog box does not contain the Run tab.

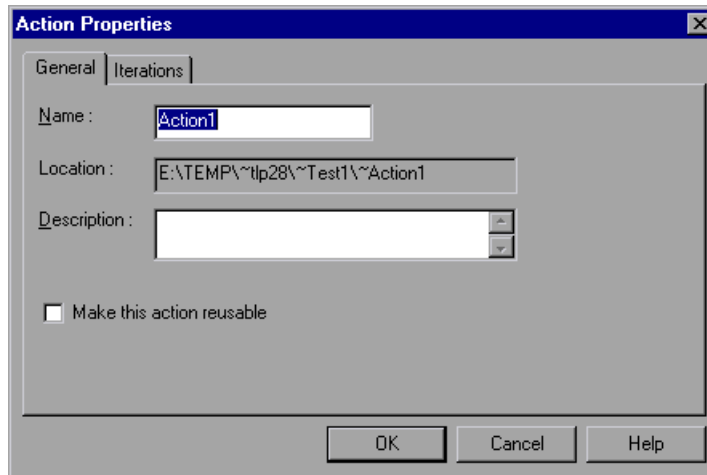


Adding or Editing an Action Description

When you add a description to an action, the description is displayed in the action's Action Properties dialog box. An action description helps you and other testers know what a specific action does without reviewing the entire script or expanded tree of the action. The description is also displayed in the description section of the Insert Copy of Action and Insert Call to Action dialog boxes. This enables you and other testers to determine which action you want to insert from another test without having to open it. For more information about inserting copies of actions and calls to actions, see "Inserting Existing Actions".

To add or edit an action description:

- 1 Right-click the action in the test tree or anywhere in the Expert View and select **Action Properties**, or choose **Step > Action Properties**. The Action Properties dialog box opens.



- 2 Enter or modify the description of the action in the **Description** box.

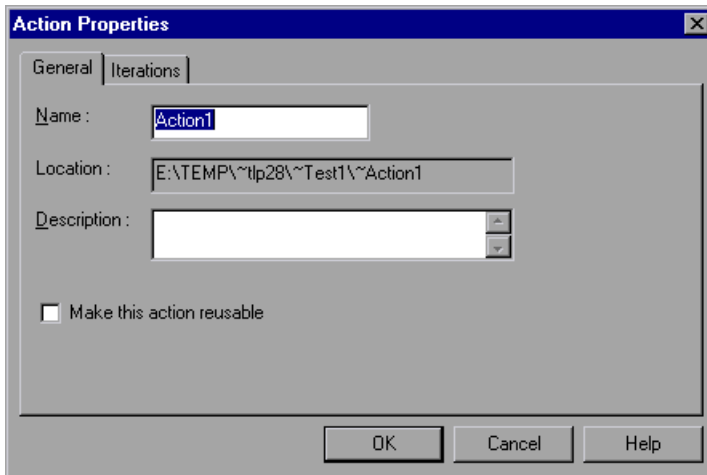
Note: You can also add a description when inserting a new action. For more information about adding a new action, see “Creating New Actions”.

Setting the Reusability Status of an Action

A reusable action can be called multiple times within a test, and can be called from other tests. Non-reusable actions can be copied and inserted as independent actions, but cannot be inserted as calls to the original action.

To change an action's reusability status:

- 1 Right-click the action in the test tree or anywhere in the Expert View and select **Action Properties**, or choose **Step > Action Properties**. The Action Properties dialog box opens.



- 2 Select or clear the **Make this action reusable** check box.

Note: If the test contains more than one call to the action, the **Make this action reusable** option cannot be cleared. If you want to make the action non-reusable, remove the additional calls to the action.

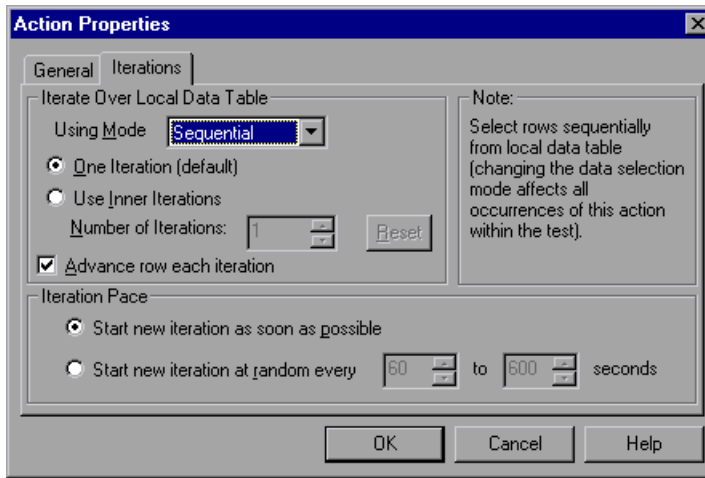


- 3 Click **OK**. If the action tree was expanded, it collapses, and the action icon changes to a reusable or non-reusable action icon, as appropriate.

Note: You cannot expand reusable actions from the test flow view. You can view details of a reusable action in the Tree View by switching to the action tree for that action. For more information about the test flow and action trees, see “Using the Action Toolbar”.

Setting the Iterations Properties for an Action

You can use the Action Properties Run tab to instruct the Virtual User Recorder to run only one iteration on the action, to run iterations on all rows in the data table, or to run iterations for only certain rows in the data table.



The Iterations tab includes the following options:

Option	Description
Iterate Over Local Table	Runs the action using all the values in the local data table.
Using Mode	<p>Specifies how the data is assigned for each iteration. Sequential assigns the data values in a sequential order. Random assigns the data values in a random order. Unique assigns a unique value for each iteration.</p> <p>The mode setting is only relevant when running the scenario in the Controller. While running the test in the Virtual User Recorder, parameters are taken from the Data table in the order in which they appear.</p>

Option	Description
No Inner Iterations	Runs all actions with the same number of iterations.
Use Inner Iterations	Allows you to specify the number of iterations to run for each action.
Advance row each iteration	Specifies whether to use, or not use, a new value for each local iteration.
Iteration Pace	Specifies how long the replay should wait between iterations.

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.

Additionally, when running a test in the Virtual User Recorder, if the number of iterations specified here exceeds the number of rows in the data table, the test will run according to the number of rows in the data table.

Removing Actions From a Test

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

- When you remove a non-reusable action, you delete the action entirely.
- When you remove a call to a reusable action, you remove the action from your test flow, but the action remains part of the test.
- When you remove a reusable action, you delete the action entirely. This will cause any test calling this action to fail.

Removing a Non-Reusable Action

Removing a non-reusable action from your test deletes the action entirely.

To remove a non-reusable action

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

Removing a Call to a Reusable Action From the Test Flow

You should choose to remove a call to a 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 sheet for the action remains. You can still view the action (and edit a reusable action) by selecting it from the Action List in the Tree View or in the Expert View.

After you remove a call to an action, you can insert the action back into the test from which it was removed, or into any other test using the **Insert Copy of Action** or **Insert Call to Action** options. For more information see “Inserting Existing Actions”.

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**, 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 Highlight the action icon in the test tree.
- 3 Choose **Edit > Delete**, or press the **Delete** key on your keyboard. The Confirm Action Removal message box opens.
- 4 Click **Yes** to close the message box and remove the call to the action.

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 parameters and local (action-specific) parameters. For example, you can create a test in which a travel agent logs into the flight reservation system, books three flights, and logs out; the next travel agent logs into the flight reservation system, books three flights, and logs out, and so on. To parameterize the 'book a flight' action, you choose **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 detailed action descriptions. This facilitates inserting actions from one test to another. You can rename an action by choosing **Edit > Rename Action**.

- If you plan to use an identical or virtually identical procedure in more than one test, you should consider inserting an action from another test. If you want to make slight modifications to the action in only one test, you should use the **Insert Copy of Action** option to paste a copy of the action. If you want modifications to affect all tests containing the action, you should use the **Insert Call to Action** option to insert a link to the action in the original test.
- Reusable actions help you to maintain your tests, but it is important to consider the effects of making an action reusable. Once you make an action reusable, be sure to consider how changes to your action could potentially affect other tests that call that action.
- If your test is part of a TestDirector project, all actions in the test should also be stored in a TestDirector project. If your test is saved in the file system, all actions in the test should also be saved in the file system.
- If you expect other users to open your tests, and all actions in your tests are in the same drive, you should use relative paths for your reusable actions, so that other users will be able to open your tests even if they have mapped their network drives differently. When working with TestDirector tests and actions, you must use absolute paths.
- 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 13, “Working with Data Tables.”

13

Working with Data Tables

The Virtual User Recorder 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 Methods

About Working with Data Tables

You can insert input and output parameters into your test so that it will run several times on different sets of data. Each test run on a different set of data is called an *iteration*. The data your test uses is stored in the data table, which is displayed in the Data pane at the bottom of the screen. The data table has the characteristics of a Microsoft Excel spreadsheet, meaning that you can also execute mathematical formulas within the cells.

After you run a test, the data you enter in the data table is displayed in the Runtime data table within the Test Results window. For more information on the Runtime data table, see Chapter 16, “Analyzing Test Results in Stand-Alone Mode.”

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, the Virtual User Recorder would use the data contained in the Purchase sheet when running iterations on steps with local parameters within the *Purchase* action.

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

For more information about creating global and local parameters, see Chapter 7, “Parameterizing Tests.”

Editing the Data Table

The data table contains the values that Astra LoadTest substitutes for parameters when you run a test. Whenever you save your test, Astra LoadTest automatically saves the test’s data table as an *.xls* file.

By default, the data table is saved in the test folder. You can save the data table in another location (and instruct the test to use this data table when running the test) by specifying it in the Properties tab of the Test Settings dialog box. This can be useful if you want to run the same test with different sets of input values. For example, you can test the localization capabilities of your application by running your test with a different data table file for each language you want to test. For more information on the Test Settings dialog box, see Chapter 23, “Setting Testing Options for a Single Test.”

Tip: You can also vary the user interface strings that you check in each language by using a different environment parameter file each time you run the test. For more information, see Chapter 7, “Parameterizing Tests.”

You can edit information in the table by typing directly into the table. You can also import data in Excel 95, Excel 97, Excel 2000, or ASCII format. You use the table in the same way as an Excel spreadsheet, including inserting formulas into cells.

To edit the data table:

- 1 Open your test.
- 2 Make sure the **Data Views** button is enabled.



	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 LoadTest 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.

Edit Command	Description
Paste Values	Pastes values from the Clipboard to the current table selection. Any formatting applied to the values is ignored. In addition, only formula results are pasted; formulas are ignored.
Clear	Clears formats or contents from the current selection. You can clear only formats, only contents (including formulas), or both formats and contents.
Insert	Inserts empty cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells.
Delete	Deletes the current selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells.
Fill Right	Copies data in the left-most cell of the selected range to all cells to the right of it within the selected range.
Fill Down	Copies data in the top cell of the selected range to all cells below it within the selected range.
Find	Finds a cell containing specified text. You can search by row or column in the table and specify to match case or find entire cells only.
Replace	Finds a cell containing specified text and replaces it with different text. You can search by row or column in the table and specify to match case and/or to find entire cells only. You can also replace all.
Go To	Goes to a specified cell. This cell becomes the active cell.

Data Menu

The following commands are available in the Data menu:

Data Command	Description
Recalc	Recalculates any formula cells in the table.
Sort	Sorts a selection of cells by row and/or column and keys.

Data Command	Description
AutoFill List	Creates, edits, or deletes an autofill list. An autofill list contains frequently-used series of text such as months and days of the week. When adding a new list, separate each item with a semi-colon. To use an autofill list, enter the first item into a cell in the table. Drag the cursor, from the bottom right corner of the cell, and the Virtual User Recorder 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: (dynamic)	Sets format to Date with the M/d/yy format.
Time: h:mm AM/PM	Sets format to Time with the h:mm AM/PM format.

Format Command	Description
Custom Number	Sets format to a custom number format that you specify.
Validation Rule	Sets validation rule to test data entered into a cell or range of cells. A validation rule consists of a formula to test, and text to display if the validation fails.

Importing Data from a Database

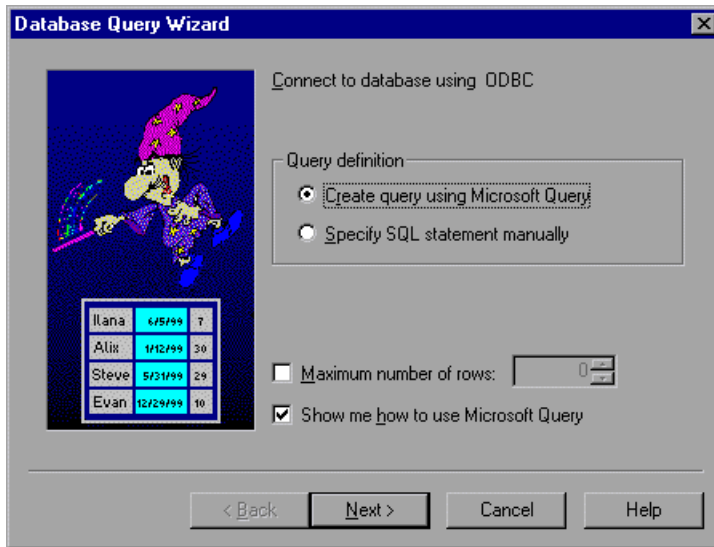
You can import data from a database by selecting a Query from Microsoft Query or by manually specifying an SQL statement.

You can install Microsoft Query from the *custom installation* of Microsoft Office.

Note: Contrary to importing an excel file (**File > Import > From File**), existing data in the data table is *not* replaced when you import data from a database. If the database you import contains a column with the same name as an existing column, the database column is added as a new column with the column name followed by a number. For example, if your data table already contains a column called departures, a database column by the same name would be inserted into the data table as: departures1.

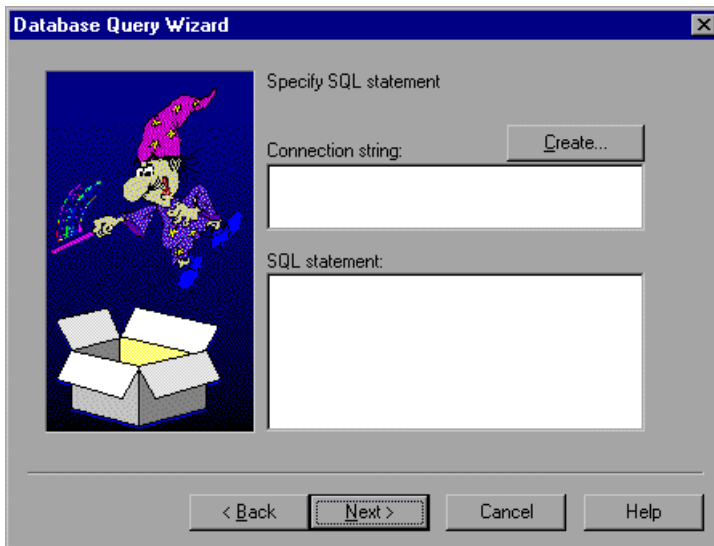
To import data from a database:

- 1 Right-click on the data table sheet to which you want to import the data and select **File > Import > From Database**. The Database Query Wizard opens.



- 2 Select your database selection preferences and click **Next**. You can choose from the following options:
 - **Create query using Microsoft Query:** Opens Microsoft Query, enabling you to create a new query. Once you finish defining your query, you are exit back to the Virtual User Recorder.
 - **Specify SQL statement manually:** Opens the **Specify SQL statement** screen in the wizard, which enables you to specify the connection string and an SQL statement. For additional information, see step 3.
 - **Maximum number of rows:** Select this check box and enter the maximum number of database rows to check. You can specify a maximum of 32,000 rows.
 - **Show me how to use Microsoft Query:** Displays an instruction screen before opening Microsoft Query, when you click **Next**. (Enabled only when **Create query using Microsoft Query** is selected).
- 3 If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information about creating a query, see “Creating a Query in Microsoft Query”.

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



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

- **Connection string:** Enter the connection string, or click **Create** to open the ODBC Select Data Source dialog box. You can select a *.*dsn* file in the ODBC Select Data Source dialog box to have it insert the connection string in the box for you.
 - **SQL statement:** Enter the SQL statement.
- 4** The Virtual User Recorder takes several seconds to capture the database query and restore the Virtual User Recorder 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. The Virtual User Recorder 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 LoadTest** to close Microsoft Query and return to the Virtual User Recorder.
 - ▶ Version 8.00 and Version 2000: In the Finish screen of the Query Wizard, select **Exit and return to Astra LoadTest** 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 LoadTest** to close Microsoft Query and return to the Virtual User Recorder.
- 4 Return to step 4 in “Importing Data from a Database” to continue creating your database checkpoint in the Virtual User Recorder.

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

Using Formulas in the Data Table

You can use any Microsoft Excel formula in your data table. This enables you to create contextually relevant data during the test run. You can also use formulas as part of a checkpoint to check that objects from a page created on-the-fly (dynamically generated) or other variable objects in your Web page have the values you expect for a given context.

When you use formulas in a data table to compare values (generally in a checkpoint), the values you compare must be of the same type, i.e. integers, strings, etc. When you extract values from different places in your applications using different functions, the values may not be of the same type. Although these values may look identical on the screen, a comparison of them will fail, since, for example, 8.2 is not equal to “8.2”. You can use the **TEXT** and **VALUE** functions to convert values from one type to another:

- ▶ **TEXT(value)** returns the textual equivalent of a numeric value, so that, for example, `TEXT(8.2)="8.2"`;
- ▶ **VALUE(string)** returns the numeric value of a string, so that, for example, `VALUE("8.2")=8.2`.

For additional information on using worksheet functions, refer to the *Microsoft Excel* documentation.

Using Formulas to Create 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

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 LoadTest 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:

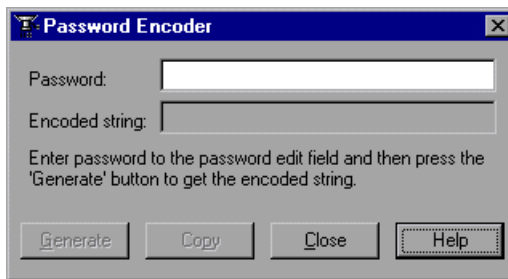
- 1 Select the page, text, or object for which you want to create a checkpoint and open the Insert Checkpoint dialog box as described in Chapter 5, "Creating Checkpoints."
- 2 Click Parameter and specify a logical name for the parameter.
- 3 Select the User data table formula check box.
- 4 Specify your other checkpoint setting preferences as described in Chapter 5, "Creating Checkpoints."
- 5 Click **OK**. The two columns are added to the table, and a checkpoint  icon is added to your test tree.
- 6 Highlight the value in the first (formula) column to view the formula and modify the formula to fit your needs.
- 7 If you want to run several iterations, add the appropriate formula in subsequent rows of the formula column for each iteration in the test or action.

Inserting Encoded Passwords into the Data Table

You can encode passwords in order to use the resulting strings as parameter values. For example, your Web site may include a form in which the user must supply a password. You may want to test how your site responds to different passwords, but you also want to ensure the integrity of the passwords. The **Password Encoder** enables you to encode your passwords and place secure values into the data table.

To encode a password:

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



- 2 Enter the password in the **Password** box.
- 3 Click **Generate**. The Password Encoder encrypts the password and displays it in the **Encoded String** box.
- 4 Use the **Copy** button to copy and paste the encoded value in the data table.
- 5 Repeat the process for each password you want to encode.
- 6 Click **Close** to close the Password Encoder.

Using Data Table Scripting Methods

Astra LoadTest provides several data table methods that enable you to retrieve information about the data table and to set the value of cells in the data table.

You enter these statements manually in the Expert View. For more information about working in the Expert View see Chapter 20, "Testing in the Expert View."

From a programming perspective, the data table is made up of three types of objects: DataTable, Sheet (sheet), and Parameter (column). Each object has several methods and properties that you can use to retrieve or set values.

For more details on the data table methods, refer to the *Astra LoadTest Function Reference*.

14

Handling Unexpected Events and Errors

You can instruct the Virtual User Recorder to handle unexpected events and errors that occur in your testing environment during a test run.

This chapter describes:

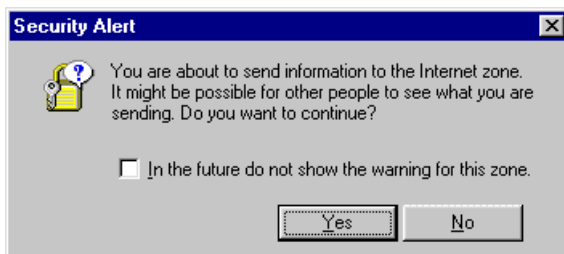
- Changing the Status of Exceptions
- Modifying Exceptions
- Adding New Exceptions
- Deleting Exceptions
- Configuring Event Handling

About Handling Unexpected Events and Errors

Unexpected events and errors during a test run can disrupt your test and distort test results. This is a problem particularly when running tests unattended: the tests are suspended until you perform the action needed to recover.

You can use the *Exception Editor* to instruct the Virtual User Recorder 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 the Virtual User Recorder 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 the Virtual User Recorder 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 the Virtual User Recorder 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 15, “Running Tests in Stand-Alone Mode.”

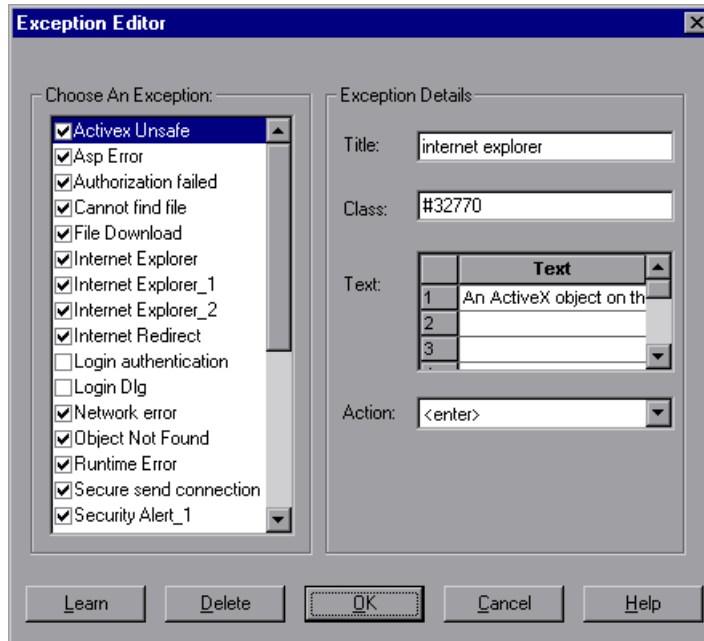
The Virtual User Recorder stores information about handling unexpected events and errors in an external exception configuration file. You can use this file to “learn” exceptions on one machine and “teach” them to other machines. You can also use this file to handle exceptions when testing localized versions of a single application.

Changing the Status of Exceptions

The Exception Editor includes a list of all the available exceptions. You can choose to activate or deactivate any exception in the list.

To change the status of an exception:

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



- 2 In the **Choose An Exception** list, click an exception.

The exception is highlighted. The current description of the exception appears in the Exception Details area.

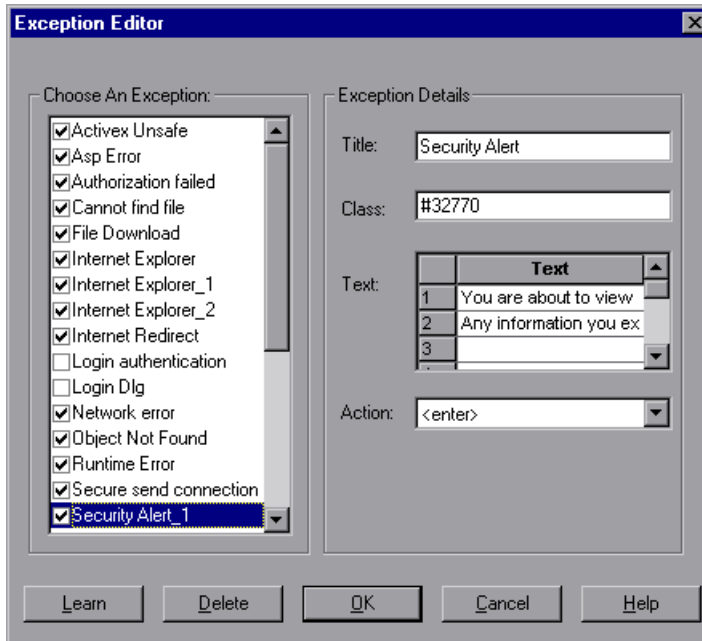
- 3 To activate an exception, select its check box. To deactivate the exception, clear its check box.
- 4 Click **OK** to save the changes.

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.

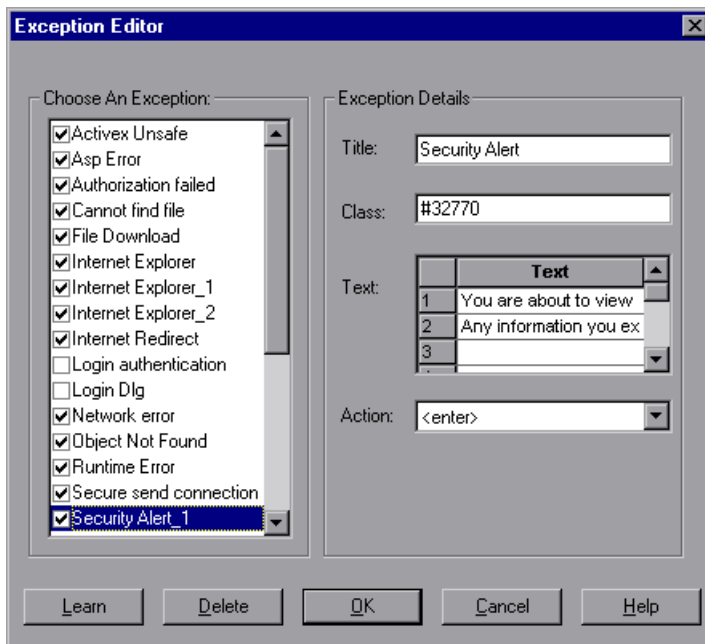
- 4** Click **OK** to save the changes.

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.
- 2 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 208.

- 3 Click **OK** to save the exception.

Deleting Exceptions

You can delete an exception from the list of exceptions in the Exception Editor.

To delete an exception:

- 1** Choose **Tools > Exception Editor**. The Exception Editor opens.
- 2** In the **Choose An Exception list**, click an exception to delete.
The exception is highlighted. The current description of the exception 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.

Configuring Event Handling

The Virtual User Recorder stores information about handling unexpected events and errors in an external exception configuration file. This file, *Exception.inf*, is located in the *[Astra LoadTest installation]\dat* folder.

You may want to access this file for the following reasons:

- to “learn” exceptions on one machine and “teach” them to other machines

Tip: Set the exception handling as described in this chapter on one machine. Then replace the file on the other machine(s) with the new exception file.

- to handle exceptions when testing localized versions of a single application

Tip: Set the exception handling as described in this chapter for each localized version. When you change the localized version of the application you are testing, you simply switch the *Exception.inf* exception configuration file.

Part III

Running and Debugging Tests

15

Running Tests in Stand-Alone Mode

In order to perform load testing with a recorded test, you use the Controller to run the test within a scenario. Before integrating the test into a scenario, you check its functionality by running the test in stand-alone mode.

Running the test in stand-alone mode means running the test without using the Controller. This is done to establish how the test will execute when run from the Controller.

This chapter describes:

- Choosing a Run Mode
- Running a Test
- Using Optional Steps

About Running Tests

When you run a test, the Virtual User Recorder navigates through your Web site, performing the steps you recorded. If your test does not contain parameterized values, the Virtual User Recorder runs the test once. If the test contains global parameters (in the Global tab of the Data pane) or local parameters (in an Action tab of the Data Pane), the Virtual User Recorder runs the test as specified in the Iterations tab of the Test Settings and the Action Properties dialog boxes. For additional information, see Chapter 23, “Setting Testing Options for a Single Test” and Chapter 12, “Working with Actions.”

Once the test run is complete, a Debug mode replay displays a report detailing the test results, while a Load mode replay details any errors in the Log Viewer and the Run-Time Viewer.

Note: Astra LoadTest runs the HTML scripts on the client side and supports both JavaScript and VBScript. This allows you to run your test without correlating it first.

You can also run tests on objects with dynamic descriptions.

Choosing a Run Mode

The Virtual User Recorder provides both a Debug and Load run mode. Each run mode uses the same test and repository, so you only need to record the test once. The main difference between the two modes is the run-time replay level.

Both the Debug mode and Load mode always run a test from the first step in the test.

Debug mode runs your test in a browser and provides debugging capabilities.

Load mode runs your test as if it is part of a load scenario in the Controller, and does not provide debugging capabilities. A test that passes a Load mode run will run successfully in the Controller. Before running in this mode, a test must first be saved.

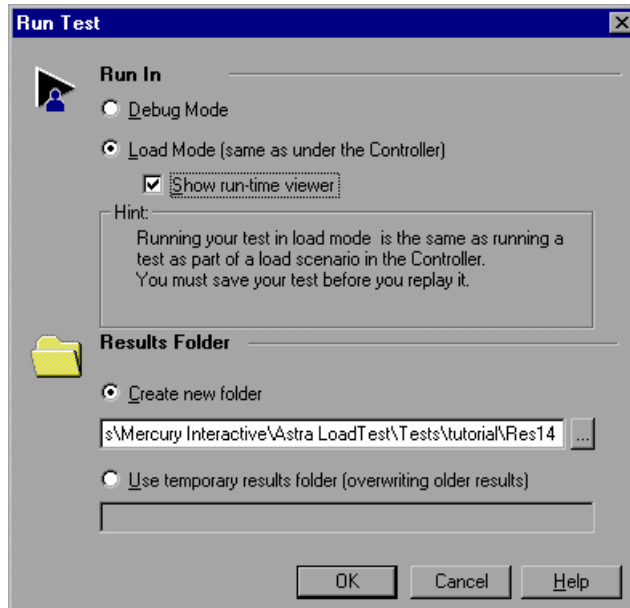
Running a Test

After recording your test and deciding on a replay mode, you are ready to run your test.

To run a test using the Load replay mode:



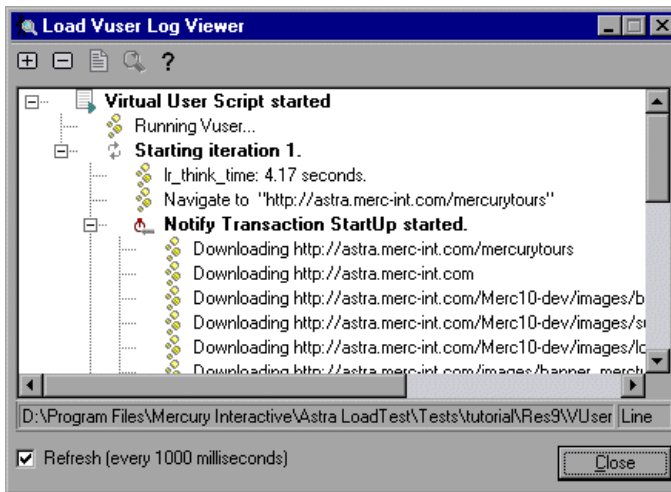
- 1 Save your test. Choose **File > Save** or click the **Save** button to save the test.
- 2 Click the **Run** button on the toolbar, or choose **Test > Run**. The Run Test dialog box opens, displaying the two run modes and a default path and test file run name for the test run results.



- 3 Select **Load Mode** and decide whether you want to enable the run-time viewer.
- 4 To run the test and save it to a new folder select the **Create new folder** option. Use the default name, type a new name in the text box, or click the browse button to locate the folder.

To run the test and overwrite the previous test run results, click **Use temporary results folder**.

5 Click **OK**. The Load Vuser Log Viewer opens.



The log contains run-time information about the Vuser and is refreshed every 1000 milliseconds. To disable the refreshing of the log, clear the **Refresh** check box.



To view the information in text format, click the **Text View** button.



To view a snapshot of the Web page where an error occurred, highlight the error in the log and double-click or click the **Display** button. The Run-Time Viewer opens.

Using the Run-Time viewer provides visual insights into where your test may be experiencing problems.



To expand the tree view so that you can view all of the run-time information about the Vuser, click the **Expand Tree** button.



To collapse the tree view, click the **Collapse Tree** button.

Note: If you close the Log viewer during the replay and would like to see it again select **Test > Show Load Mode Log Viewer**.

If you selected the option, the Run-Time Viewer opens. The viewer is not a browser, and only displays snapshots of the pages that are returned to the Vuser.



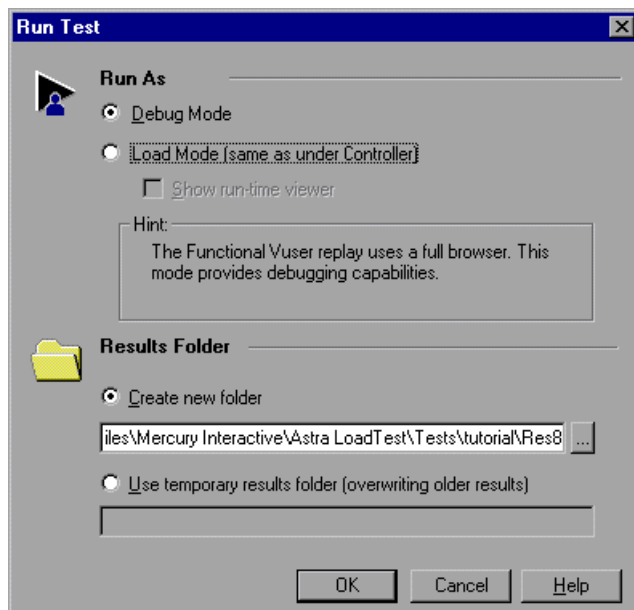
To run a test in the Debug Mode:



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 two run modes and a default path and test file run name for the test run results.



3 Select **Debug Mode**.

4 To run the test and save it to a new folder select the **Create new folder** option. Use the default name, type a new name in the text box, or click the browse button to locate the folder.

To run the test and overwrite the previous test run results, click **Use temporary results folder**.

5 Click **OK**. The Run Test dialog box closes and the Virtual User Recorder begins running the test. The Virtual User Recorder always runs a test from the first step in the test. As the Virtual User Recorder runs the test, it highlights each step in the test tree.

When the test stops running, the Test Results window opens unless you have cleared the **View results when test run ends** check box in the Run tab of the Options dialog box. For more information about the Options dialog box, see Chapter 22, “Setting the Virtual User Recorder 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 succeed in opening a particular dialog box, the Virtual User Recorder does not necessarily interrupt the test run. It can bypass the step and continue to run the test. The Virtual User Recorder will bypass any *optional* step. By default, the Virtual User Recorder 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 14, “Handling Unexpected Events and Errors.”


Setting Optional Steps

When running a test, you can bypass certain steps if they are not required, by setting them as optional steps. For example, when recording a test, the site you are testing may prompt you to enter your user name and password in a login window. When you run the test, however, the site does not

prompt you to enter your user name and password, because it has retained the information that was previously entered. In this case, the steps that were recorded for entering the login information are not required and should be marked as optional.

When running a test, if a step in an optional dialog box does not open, the Virtual User Recorder automatically bypasses this step and continues to run the test. When the test run is completed, a message is displayed for the step that failed to open the dialog box, but the step does not cause the test to fail.

To set an optional step:

Right-click a step in the test tree and choose **Optional Step**. The Optional Step  icon is added next to the selected step.

Note: You can also add an optional step from the Expert View by adding `OptionalStep` to the beginning of the VBScript statement. For example:

```
OptionalStep.Browser("browser_name").Page("page_name").Link("link_name")
```

For information on working in Expert View, see Chapter 20, “Testing in the Expert View.” For information on the **OptionalStep** object, refer to the the Virtual User Recorder *Function Reference*.

Default Optional Steps

The Virtual User Recorder 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

16

Analyzing Test Results in Stand-Alone Mode

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
- Reporting Defects Detected During a Test Run

About Analyzing Test Results

After you run your test, the test results are displayed in the Test Results window. This window contains a description of every step performed during the test run. If the test does not contain parameterized values, the Test Results window shows a single test iteration result. If the test does contain parameters, and the Run Properties for the action(s) and/or the test are set to run more than one iteration, the Test Results window shows a test iteration for each row in the table in the Data pane.

The Test Results Window

After a test run, you can view the results in the Test Results window. By default, the Test Results window automatically opens when a test run is completed. For more information on opening the Test Results window, see “Viewing the Results of a Test Run” on page 227.

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

NUM

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 307.

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.

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

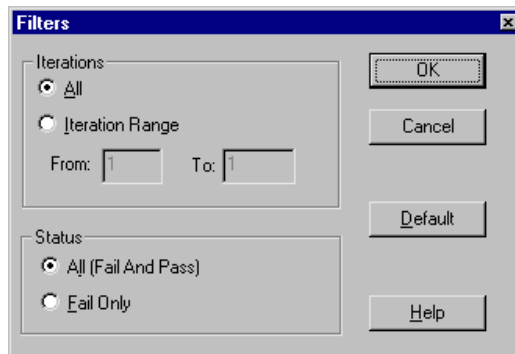
NUM

- 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 **Filters** button or choose **View > Filters**. The Filters dialog box opens.



The default filter options are displayed above.

- ▶ Click **Iteration Range** to limit the test results to a specified range of test iterations.
- ▶ Click **Fail Only** to limit the test results to test iterations that failed.



5 To view other test run results, click the **Open** button or choose **File > Open**. 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, click the **Print** button or choose **File > Print**. For additional information, see “Printing Test Results” on page 233.

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 LoadTest > Tools > 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 LoadTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails. You can view the results of the checkpoint in the Test Results window.


For more information on checkpoints, see Chapter 5, “Creating Checkpoints.”

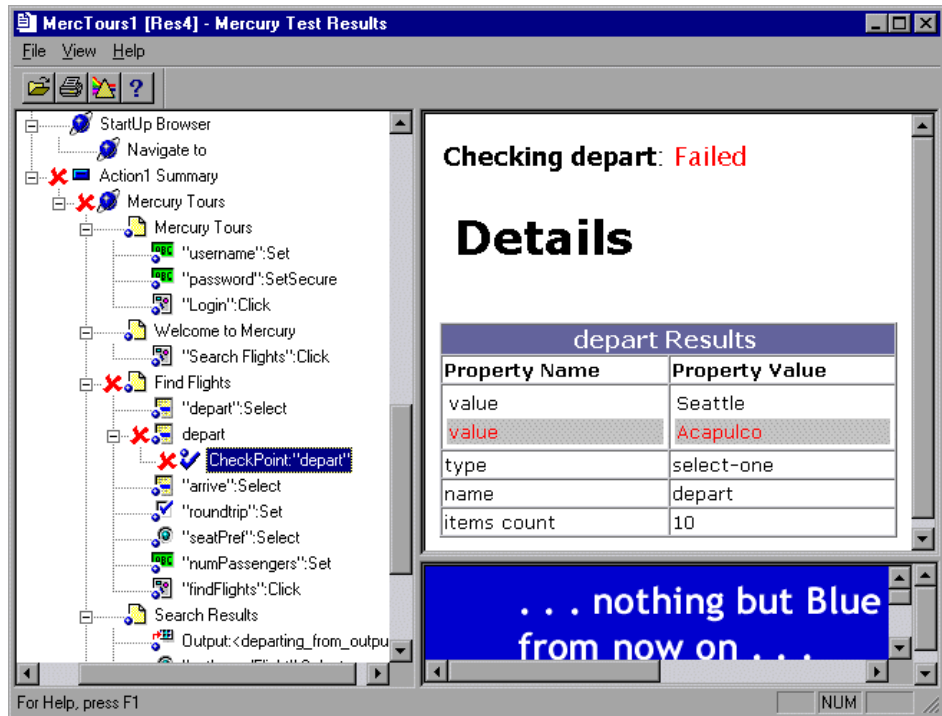
To view the results of a checkpoint:



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.



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 parameterization, see Chapter 7, “Parameterizing Tests.” For more information on output parameterization, see Chapter 10, “Creating Output Parameters.” For more information on the test Data Table, see Chapter 13, “Working with Data Tables.”

To view the Runtime data table:



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.



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

Sample [Res3] - Mercury Test Results

File View Help

Test Test1 Summary

- Runtime Data
- Test Iteration 1
- Test Iteration 2
- Test Iteration 3

	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					

Global Action1 /

For Help, press F1

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

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

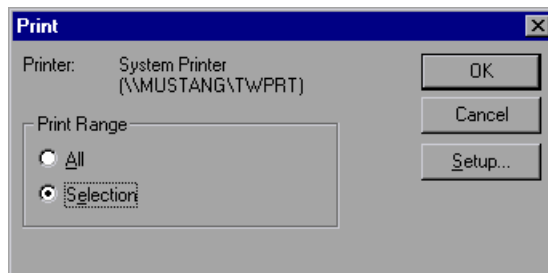
Printing Test Results

You can print your test results from the Test Results window.

To print the test results:



- 1 To print the report, click the **Print** button or choose **File > Print**. The Print dialog box opens.



- 2 Select a **Print Range** option:
 - ▶ Select **All** to print the entire results report.
 - ▶ Select **Selection** to only print a selected branch in the report tree.
- 3 Click **OK** to print.

Reporting Defects Detected During a Test Run

If a test run detects a defect, you can report it to a TestDirector project directly from the Test Results window.

To report a defect to TestDirector:



- 1 Choose **Tools > TestDirector Connection** or click the **TestDirector Connection** button to connect to a TestDirector project. For additional information, see Chapter 26, “Working with TestDirector.”



- 2** Choose **Tools > Add Defect** or click the **Add Defect** button to open the Add Defect dialog box in the specified TestDirector project. For additional information, refer to the *TestDirector User's Guide*.

17

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 Virtual User Recorder 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 Virtual User Recorder window.

Pausing Test Runs



You can temporarily suspend test runs by choosing **Debug > Pause** or clicking the **Pause** button. A paused test stops running when all previously interpreted steps have been run.



To resume running a paused test, click the **Run** button or choose **Test > Run**. The test run continues from the point that you invoked the Pause command.


Setting Breakpoints

By setting a breakpoint you can stop a test run at a specific place in a test. A breakpoint is indicated by a red-colored hand in the left margin of the test window. The Virtual User Recorder pauses the test run when it reaches a breakpoint. You can examine the effects of the test run up to the breakpoint, make any necessary changes, and then continue running the test from the breakpoint.

You can use breakpoints to:

- suspend a test run and inspect the state of your site
- mark a point from which to begin stepping through a test using the Step commands


To set a breakpoint:

- 1 Click a step or a line in the test where you want the test run to stop.
- 2  Choose **Debug > Toggle Breakpoint** or click the **Toggle Breakpoint** button. The breakpoint symbol is displayed in the left margin of the Virtual User Recorder window.


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

- 2  (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 Virtual User Recorder window.

- 2  (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 Virtual User Recorder window.

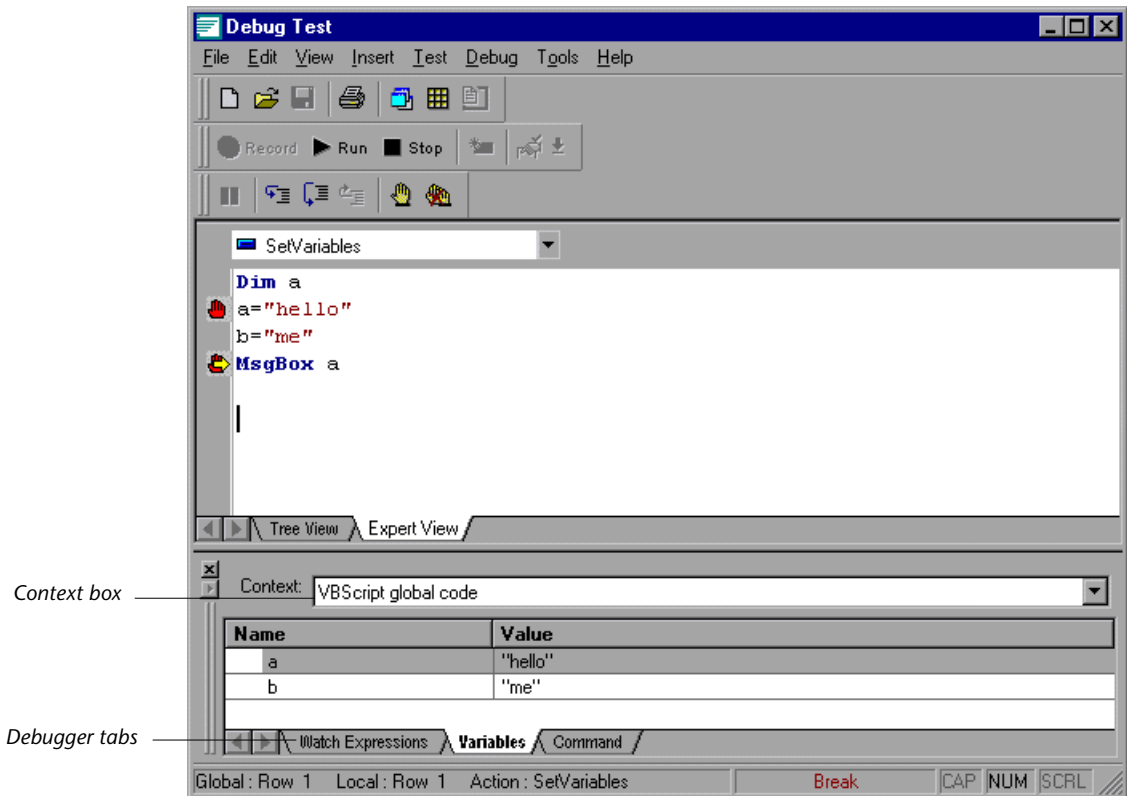
Using the Debugger Views

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

To open the Debugger pane:

- 1 Run a test with one or more breakpoints.
- 2 When the test pauses at the first breakpoint, choose **View > Debugger Views**. The Debugger pane opens at the bottom of the Virtual User Recorder 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 Virtual User Recorder window.



The Debugger tabs can display the values of variables or objects in the main script of the current action or in a selected subroutine. To switch between the main script of the action (VBScript: global code) and the subroutines and functions of the action, choose the script you want from the **Context** box.

Watch Expressions Tab

Use the Watch Expressions tab to view the current value of any variable or VBScript object that you enter in the Watch Expressions table. Paste or type the name of the object or variable into the **Name** column and press Enter to view the current value in the **Value** column. If the value of the object or variable changes when you continue to run the test, the value in the Watch Expressions tab is updated.

Note: The Virtual User Recorder updates the value of the object or variable in the Watch Expressions tab when running a test step by step.

Variables Tab

Use the Variables tab to view the current value of all variables in the current action (or selected subroutine) that have been identified up to the point where the test stopped. If the value of a variable changes when you continue to run the test, the value in the Variables tab is updated.

Note: The Virtual User Recorder updates the value of the variable in the Variables tab when running a test step by step.

Command Tab

Use the Command tab enables to execute a line of script in order to set or modify the current value of a variable or VBScript object in your test. When you continue running the test, the Virtual User Recorder 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 12, “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 20, “Testing in the Expert View.”

Step 2: Add Breakpoints

Add a breakpoint at line 2 and line 4. For more information about adding breakpoints, see “Setting Breakpoints”.

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.

Note: The breakpoint causes the test to stop before executing the corresponding line.

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

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



Click the **Run** button to continue running the test. The test stops at the next breakpoint. Note that the values of variables a and b have both been updated in the Watch Expressions and Variables tabs.

Step 6: Modify the Value of a Variable Using the Command Tab



Select the **Command** tab in the Debugger pane. Type: a="This is the new value of a" at the command prompt, and press **Enter** on the keypad. Click the **Run** button to continue running the test. The message box that appears displays the new value of 'a'.

Part IV

Advanced Features

18

Configuring Event Recording

If the Virtual User Recorder does not record all the events you need, you can configure the events you want to record for each type of Web object.

This chapter describes:

- ▶ Selecting a Standard Event Recording Configuration
- ▶ Customizing the Event Recording Configuration
- ▶ Saving and Loading Custom Configuration Files
- ▶ Resetting Standard Event Recording Configuration Settings

About Configuring Event Recording

The Virtual User Recorder 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 the Virtual User Recorder 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, the Virtual User Recorder 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 the Virtual User Recorder 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, the Virtual User Recorder uses the *Basic* recording configuration level. If the Virtual User Recorder does not record all the events you need, you may require a higher event configuration level.

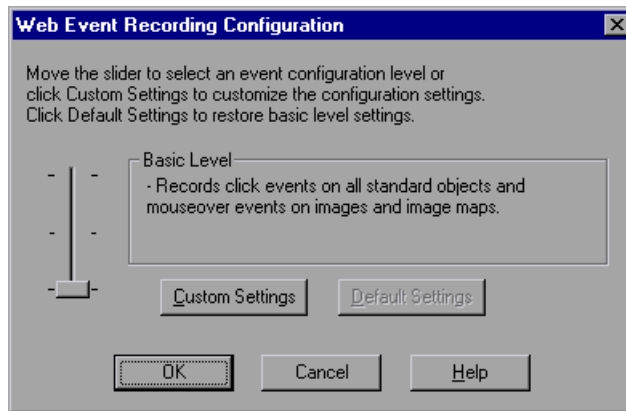
The Event Configuration Settings dialog box offers three standard event configuration levels.

Level	Description
Basic	Default <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.

Level	Description
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”.

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.

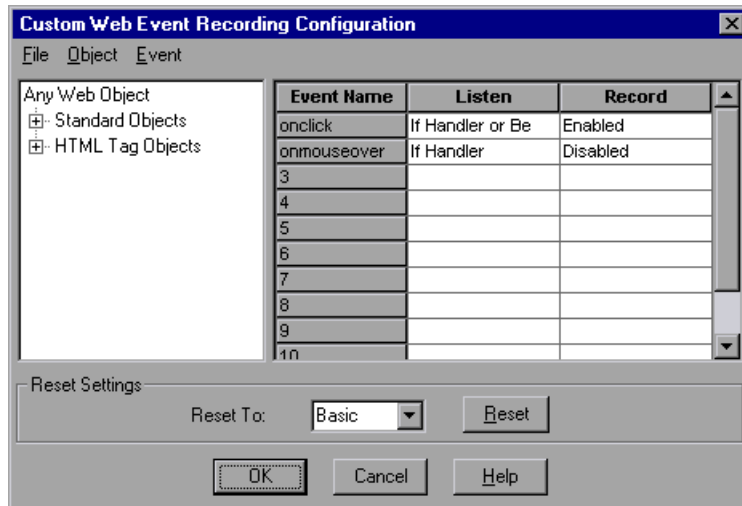
The Custom Event Configuration dialog box enables you to customize event recording in several ways. You can:

- ▶ add or delete objects to which the Virtual User Recorder should apply special listening or recording settings
- ▶ add or delete events for which the Virtual User Recorder should listen for all objects
- ▶ add or delete events for which the Virtual User Recorder should listen for one or more specific objects
- ▶ modify the listening or recording settings of an event for which the Virtual User Recorder listens for all objects
- ▶ modify the listening or recording settings of an event for one or more specific objects

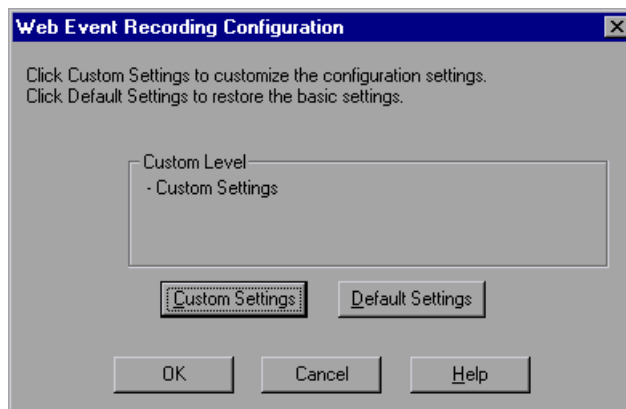
To customize the event recording configuration:

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

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



- Set the event recording configuration options you want. The sections below describe the event recording configuration options in detail.
- 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

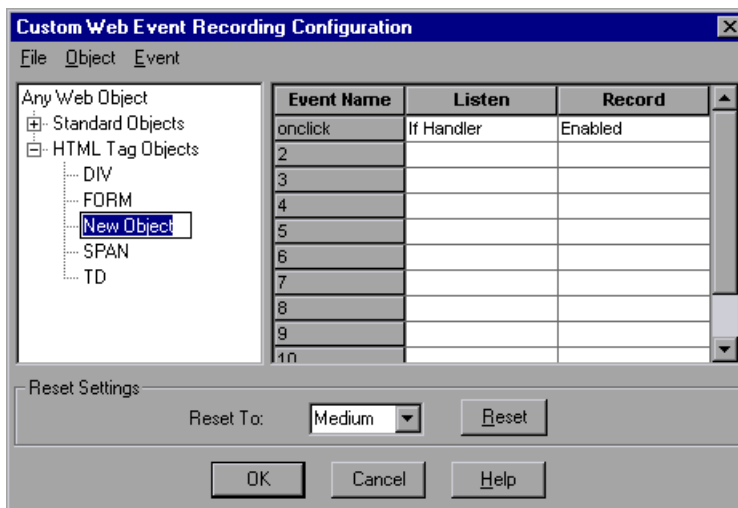
The Custom Web Event Recording Configuration dialog box lists objects in an object hierarchy. The top of the hierarchy is Any Web Object. The settings for Any Web Object apply to any object on the Web page being tested, for which there is no specific settings. Below this are the Standard and HTML Tag 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.
- ▶ The Virtual User Recorder 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”.

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

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.

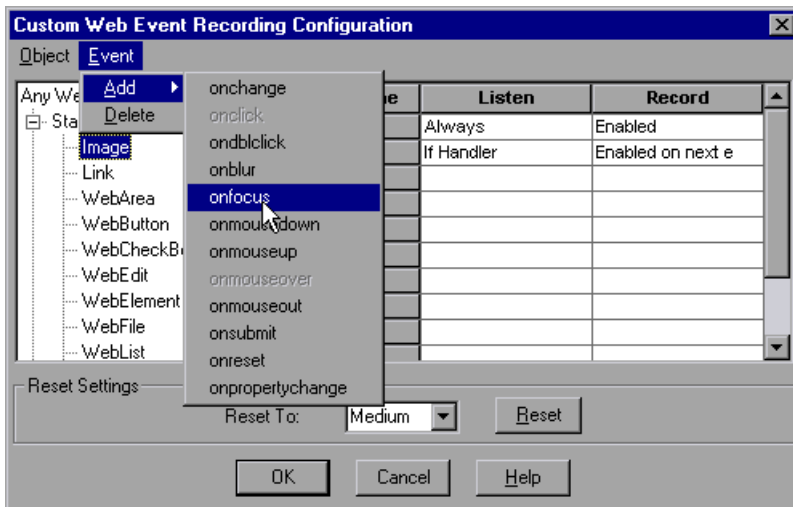
Adding and Deleting Listening Events for an Object

You can modify the list of events that trigger the Virtual User Recorder 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”.

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 the Virtual User Recorder to listen every time the event occurs on the object, if an event handler is attached to the event and/or if a DHTML behavior is attached to the event.

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

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

To specify the listening criterion for an event:

- 1 From the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the listening criterion or select **Any Web Object**.
- 2 In the row of the event you want to modify, select the listening criterion you want from the **Listen** column.

Event Name	Listen	Record
onclick	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**.

Recording Status

For each event to which the Virtual User Recorder 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**.
- 2 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		

Saving and Loading Custom Configuration Files

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

To save a custom configuration:

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

To load a custom configuration:

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

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. If you do not want to lose your changes, be sure to save your settings. For more information, see “Saving and Loading Custom Configuration Files”.

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”.

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.

19

Enhancing Your Tests with Programming

After recording a test, you can use the Virtual User Recorder 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, the Virtual User Recorder 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. The Virtual User Recorder 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.

The Virtual User Recorder 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 LoadTest 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 20, "Testing in the Expert View."

Inserting Functions

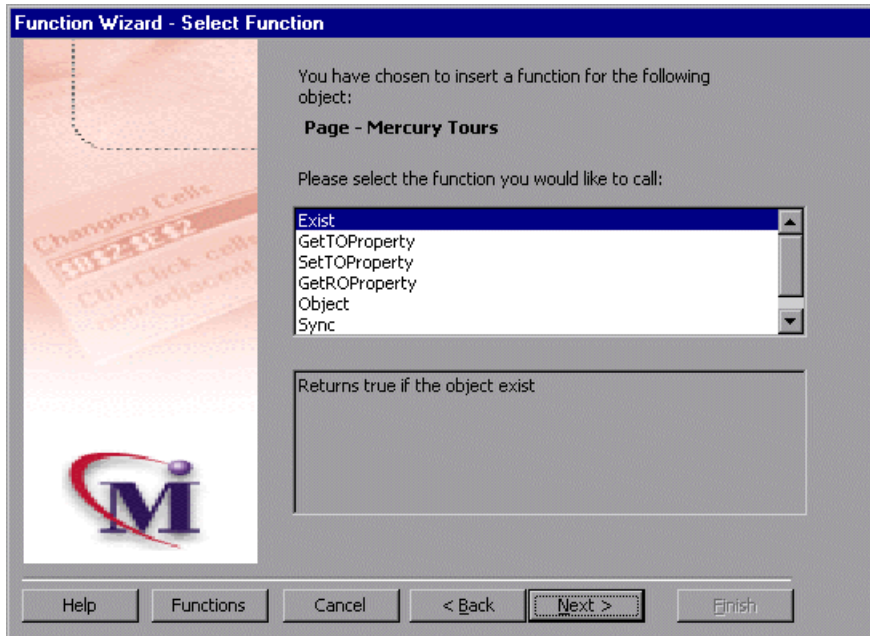
After recording, you can add additional functions to your tests using the Function Wizard. With the wizard you can add recordable and non-recordable functions that perform operations on objects or retrieve information from your site. For example, the **QueryValue** function enables you to query the method 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**.

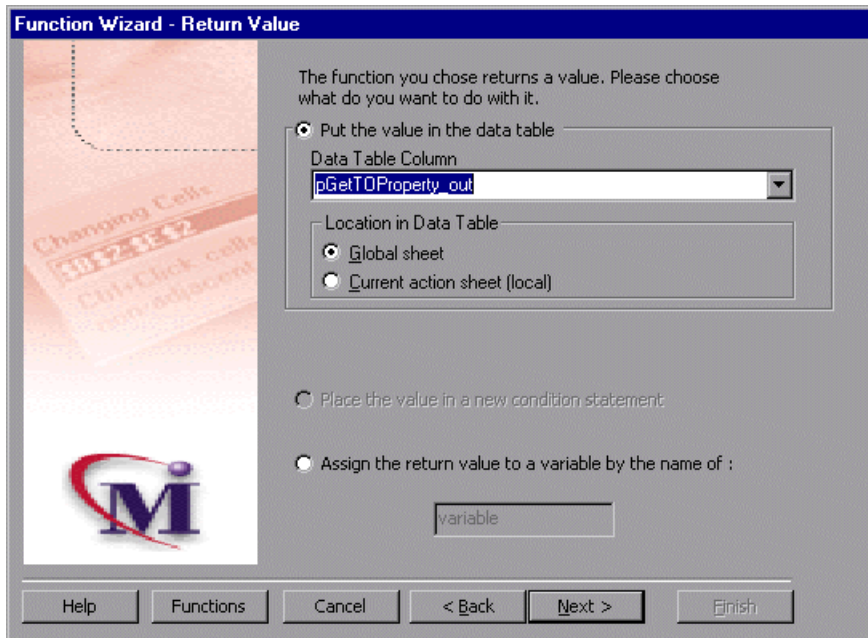
Tip: To add a function from the Expert View, click a statement in the test script. Right-click the highlighted object in the ActiveScreen and choose **Insert Function**. The **Object Selection - Object Function Wizard** opens. Select an object and click **OK**.

- 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.



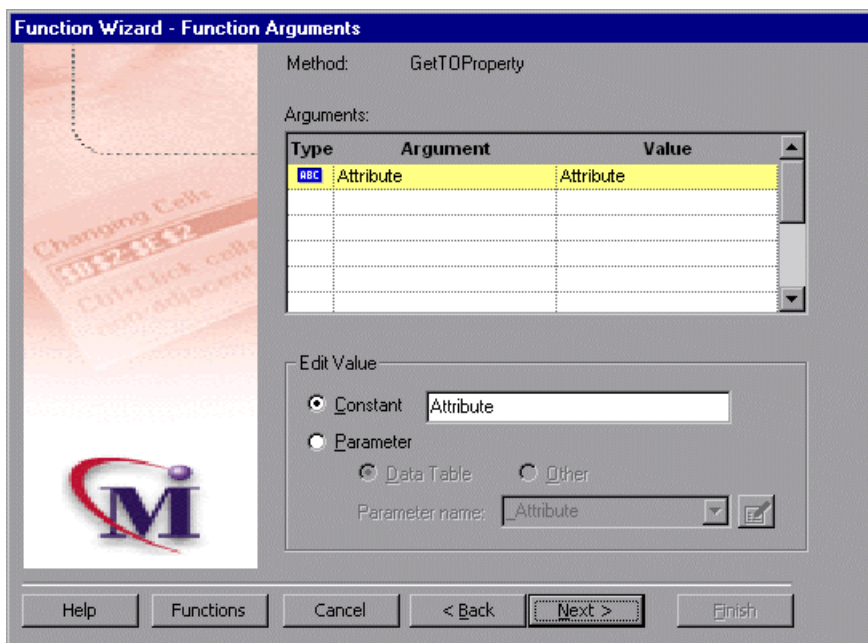
The following options are available:

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 10, "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 10, "Creating Output Parameters."



Option	Description
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 264.
Assign the return value to a variable by the name of:	Assigns the return value to a variable of the specified name.

Click **Next** to continue.

- 5 If the 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:

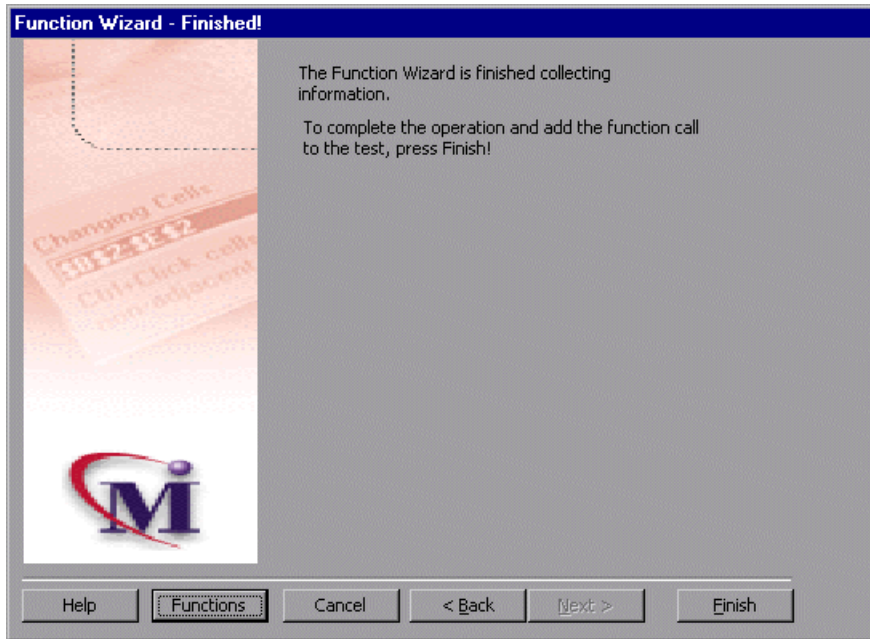
Option	Description
Type	The  icon indicates that the argument value is a constant. The  icon indicates that the argument value is a parameter.
Argument	The name of the argument whose value will be parameterized.
Value	The value of the argument to parameterize.

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 7, "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* =.

Your *If...Then...Else* statement can be nested to as many levels as you need. It has the following syntax:

```
If condition Then statements [Else elsestatements ] End If
```

Or, you can use the block form syntax:

```
If condition Then
    [statements]
    [Elseif condition-n Then
        [elseifstatements] . . .
    ]End If
[Else
    [elsestatements]
End If
```

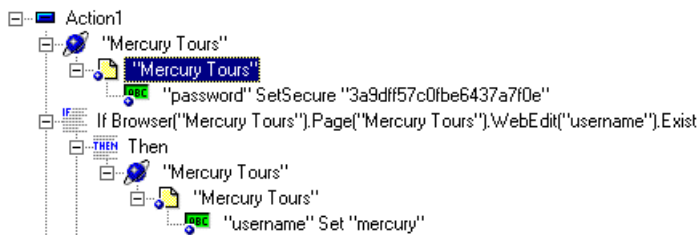
Part of Statement	Description
condition	One or more expressions that evaluate to true or false. If the condition is null, it is treated as false.
statements	One or more statements, separated by colons, that are executed if the condition is true.
condition-n	Same as <i>condition</i> .
elseif statements	One or more statements, separated by colons, that are executed if the associated <i>condition-n</i> is true.

Part of Statement	Description
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:

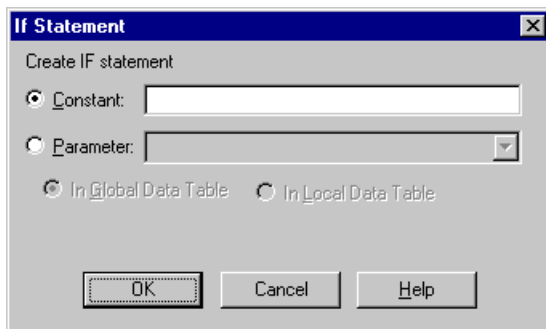


In the test tree, the following icons are used to indicate the different levels of *If...Then...Else* statements:

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 7, "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 7, "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:

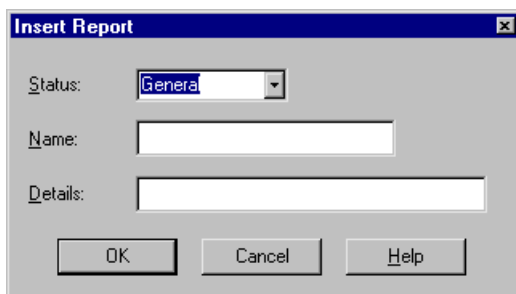
- ▶ Record a new step and then use the Cut/Paste commands to add it to your statement.
- ▶ Copy an existing step and paste it in your statement.
- ▶ Click and drag a step to move it to your statement.

Sending Messages to Your Test Results

You can define a message in your test that Astra LoadTest 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 LoadTest 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.

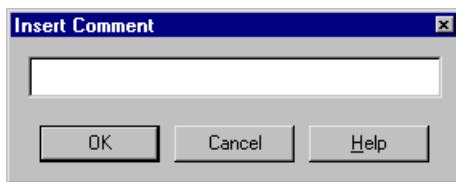
After you run the test, the  icon in the Test Results window indicates that the message was sent.

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 LoadTest 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*.

20

Testing in the Expert View

In the Virtual User Recorder, 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
- ▶ Enhancing Tests with Comments, Calculations, and Control-Flow Statements
- ▶ Retrieving and Setting Test Object Property Values
- ▶ Accessing Run-time Object Properties and Methods

About Testing in the Expert View

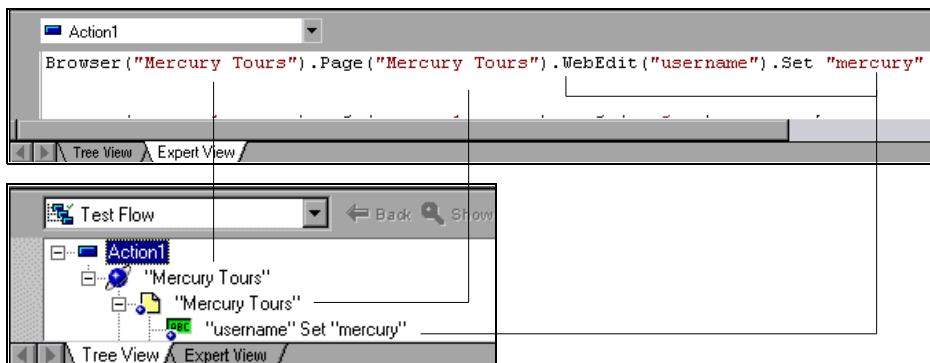
The Expert View provides an alternative to the Tree View for testers who are familiar with VBScript. In the Expert View, you can view the recorded test in VBScript and enhance it with programming.

In the Expert View you can also add methods manually instead of from the Function wizard. For information on using the Function wizard, see Chapter 4, "Enhancing Your Tests with Programming."

Understanding the Expert View

The Virtual User Recorder can display tests you record in two formats:




- ▶ (the Tree View, where the Virtual User Recorder displays the object hierarchy in an icon-based tree)
- ▶ (the Expert View, where the Virtual User Recorder 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, the Virtual User Recorder records the operations you perform on your application in terms of the objects in it. It identifies the objects in an application by specific names (e.g. a button or a list) and the method performed on the object (e.g. click or select).

To further understand how a step in the Expert View corresponds with a step in the Tree View, examine the table below:

Tree View	Expert View	Description
 "Mercury Tours"	Browser ("Mercury Tours")	The name of the Web site is "Mercury Tours".
 "Mercury Tours"	Page ("Mercury Tours")	The name of the current page in the Web site is "Mercury Tours".
 "username"	WebEdit("username")	The name of the edit field upon which the action is performed is "username".
Set "mercury"	Set "mercury"	The name of the method performed on the edit box is "Set". The name inserted into the edit box is "mercury".

An object's logical name is displayed in parentheses following the object type. In the following example, the object type is Browser, and the logical name of the Browser is "Mercury Tours":

```
Browser ("Mercury Tours")
```

The object types in the object hierarchy are separated by a period. In the following example, Browser and Page are two separate objects:

```
Browser("Mercury Tours").Page("Mercury Tours")
```

The method performed on the object always is displayed at the end of the line of script. In the following example, the word "mercury" is inserted in the "username" edit box using the **Set** method:

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

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

Checkpoints

In the Virtual User Recorder, you create checkpoints on pages, text strings, and objects. When you create a checkpoint in the Tree View, the Virtual User Recorder creates a corresponding line in VBScript in the Expert View. It uses the **Check** method to perform the checkpoint.

For example, in the following statement the Virtual User Recorder performs a check on the word “confirmed”:

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

The corresponding step in the Tree View is displayed as follows:

 Checkpoint "confirmed"

Notes:

You can only insert checkpoints into your test in the Expert View while you are recording. Use the Tree View to insert and modify checkpoints while editing your test.

The details about a checkpoint are stored with its action and are not contained in the statement displayed in the Expert View. Therefore, a checkpoint statement from the Expert View to another action or to another test.

For more information on inserting and modifying checkpoints, see Chapter 3, “Understanding Checkpoints.”

Parameters

You can use the Virtual User Recorder 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, the Virtual User Recorder creates a corresponding line in VBScript in the Expert View.

The Virtual User Recorder calls the values of a parameterized object from the Data Table using the following syntax:

Object_Hierarchy.Method **DataTable** (*parameterID*, *sheetID*)

<i>Object_Hierarchy</i>	The object-oriented definition of the parameterized object.
<i>Method</i>	The name of the method that the Virtual User Recorder 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 “dtGlobal” is displayed.

Note: You cannot create parameters from the Expert View. Use the Tree View to create parameters. For more information on parameterization, see Chapter 12, “Parameterizing TestsTransaction Files.”

For example, suppose you are creating a test on the Mercury Tours site, and you select “Paris” as your destination. The following statement is inserted into your test in the Expert View:

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

Now suppose you want to parameterize the destination, and you create a “Departure” column in the Data Table. The previous statement is modified to the following:

```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select  
    DataTable("Departure",dtGlobalSheet)
```

where **Select** is the method name, **DataTable** is the object, **Departure** is the name of the column in the Data Table, and **dtGlobalSheet** is the name of the sheet in the Data Table.

Programming in the Expert View

The Expert View displays in VBScript the steps you executed while recording your test. After you record your test, you can increase its power and flexibility by adding recordable and non-recordable VBScript statements. You can add statements that perform operations on objects or retrieve information from your site. For example, you can add a step that checks that an object exists, or you can retrieve the return value of a method.

The objects in Astra LoadTest are divided into categories. The objects include:

- ▶ Environment - functions that let you retrieve Vuser information
- ▶ Services - functions that affect the test as it runs in a load test scenario
- ▶ WebUtil - functions that enable you to set or modify Web related settings

Most objects have corresponding methods. For example, the **Back** method is associated with the Browser object.

In the following example, the user inserts “mercury” in the User Name edit box while recording. The following line is recorded in the Expert View:

```
Browser ("Mercury_Tours"). Page ("Mercury_Tours"). WebEdit ("username").  
    Set "mercury"
```

When running the test, the **Set** method sets the “mercury” text into the WebEdit object.

In the following example, the user selects “Paris” from the Departure City drop-down list while recording. The following line is recorded in the Expert View:

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

When running the test, the **Select** method selects Paris in the WebList object.

Note: For more information on Astra LoadTest objects, methods, and properties, refer to the Astra LoadTest *Function Reference*. To open the reference, choose **Help > Astra LoadTest Function Reference**.

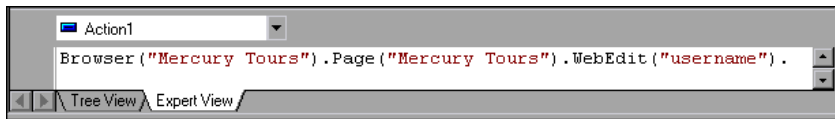
Generating a Method for an Object

In the Expert View you can generate methods. You can also generate methods in the Tree View using the Method wizard. For additional information, see Chapter 4, “Enhancing Your Tests with Programming.”

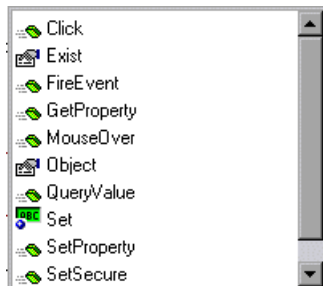
By default, the Virtual User Recorder displays the syntax for methods as you type. You can disable or enable this **Statement Completion** option in the Editor Options dialog box. For additional information, see Chapter 5, “Customizing the Expert View.”

To generate a method in the Expert View:

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

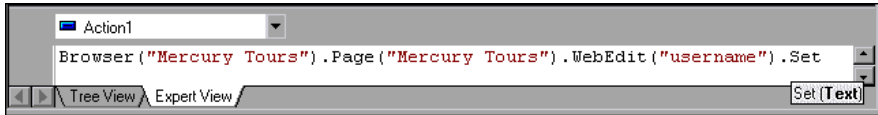


- 2 A list of the available methods for the object is displayed.



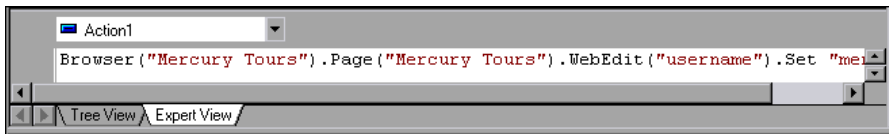
Double-click a method in the list or use the arrow keys to choose a method and click Enter. The Virtual User Recorder inserts the method into the line of script.

- 3 If the method contains arguments, and the Statement Completion option is enabled, the Virtual User Recorder displays the syntax of the method.



In the above example, the `Set` method has one argument, called *Text*. The argument name represents the text to enter in the edit box.

- 4 Insert any required arguments after the method.



Tip: When programming in VBScript, use parentheses around method arguments if the method is expected to return a value. If the method does not return a value, do not use parentheses. Arguments entered as strings must be surrounded by quotation marks.

Note: To find the syntax for a method, refer to the Astra LoadTest *Function Reference*.

Using Descriptive Programming

When you record an operation on an object, The Virtual User Recorder adds the appropriate test object to the Object Repository. Once the object exists in the Object Repository, you can add statements in the Expert View to perform additional methods on that object. To add these statements, you

usually enter the logical name of each of the objects in the object's hierarchy as the object description, and then add the appropriate method.

For example, in the statement below, “username” is the logical name of an edit field. The edit field is located on a page with the logical name “Mercury Tours” and the page was recorded in a browser with the logical name “Mercury Tours”.

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

Because each object in the Object Repository has a unique logical name, this is all you need to describe the object. During the test run, The Virtual User Recorder finds the object in the Object Repository and uses the other property values stored for that test object to identify the object in your Web site .

You can also add statements to perform methods on objects without using the Object Repository. To do this, you need to enter more information in the description of the object in order to uniquely describe the object so that The Virtual User Recorder can identify the object during the test run.

For example, suppose you recorded on a Web form in your Web site. Then, after you created your test, an additional edit field was added to the form. The Browser and Page objects already exist, but the new edit object does not. Rather than recording a new step in your existing test, you can add a statement to the script that describes the new edit object, and performs a **Set** method on it.

You describe the object by using as many *property:=value* pairs as necessary to uniquely identify the object.

The general syntax is:

```
TOClass("PropertyName1:=PropertyValue1", ... ,  
"PropertyNameX:=PropertyValueX")
```

TOClass: The test object class.

PropertyName:=PropertyValue: The test object property and its value. Each *property:=value* pair should be separated by double quotes and commas.

When you use descriptive programming, The Virtual User Recorder creates a live instance of the test object and assigns it the property values from your description. The Virtual User Recorder uses the property values in the live instance of the test object to identify the object during the test run.

The statement below creates a WebEdit test object in the Mercury Tours page with the Name author and an index of 3. When the test runs, The Virtual User Recorder finds the WebEdit object with matching property values and enters the text "Mark Twain".

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("Name:=Author",  
"Index:=3").Set "Mark Twain"
```

For more information about working with test objects, see Chapter 4, "Managing Test Objects."

If you want to use the same object several times in one test, you can assign the object you create to a variable.

For example, instead of entering:

```
Browser("index").Page("Fill-Out Form_2").WebEdit("entry1").Set "One"  
Browser("index").Page("Fill-Out Form_2").WebEdit("entry2").Set "Two"  
Browser("index").Page("Fill-Out Form_2").WebCheckBox("box1").Set "ON"  
Browser("index").Page("Fill-Out Form_2").WebCheckBox("box2").Set "ON"  
Browser("index").Page("Fill-Out Form_2").WebEdit("entry").Set "three"  
Browser("index").Page("Fill-Out Form_2").WebButton("Submit Query").Click
```

You can enter:

```
set MyPage = Browser("index").Page("Fill-Out Form_2")  
MyPage.WebEdit("entry1").Set "One"  
MyPage.WebEdit("entry2").Set "Two"  
MyPage.WebCheckBox("box1").Set "ON"  
MyPage.WebCheckBox("box2").Set "ON"  
MyPage.WebEdit("entry").Set "three"  
MyPage.WebButton("Submit Query").Click
```

Alternatively, you can use a With statement:

```
With Browser("index").Page("Fill-Out Form_2")
```



```
.WebEdit("entry1").Set "One"
.WebEdit("entry2").Set "Two"
.WebCheckBox("box1").Set "ON"
.WebCheckBox("box2").Set "ON"
.WebEdit("entry").Set "three"
.WebButton("Submit Query").Click
End With
```

For more information about the **With** statement, see “With” Statement.

Using Descriptive Programming for the WebElement Object

The *WebElement* object enables you to perform methods on Web objects that may not fit into any other Mercury test object class. The *WebElement* test object is never recorded, but you can use descriptive programming with the *WebElement* object to perform methods on any Web object in your Web site.

For example, when you run the statement below:

```
Browser("Mercury Tours").Page("Mercury
Tours").WebElement("Name:=UserName", "Index:=0").Click
```

The Virtual User Recorder clicks on the first Web object in the Mercury Tours page with the name *UserName*.

For more information about the *WebElement* object, refer to the *Astra LoadTest Function Reference*.

Using the Index Property in Descriptive Programming

The *Index* property can sometimes be a useful test object property for uniquely identifying an object. The *Index* test object property identifies an object based on the order in which it appears within the source code, where the first occurrence is 0.

Note that *Index* property values are object-specific. Thus, if you use *Index:=3* to describe a **WebEdit** test object, The Virtual User Recorder searches for the fourth **WebEdit** object in the page.

If you use `Index:=3` to describe a **WebElement** object, however, The Virtual User Recorder searches for the fourth Web object on the page regardless of the type, because the **WebElement** object applies to all Web objects.

For example, suppose you have a page with the following objects:

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

The description below refers to the third item in the list above, as it is the first WebEdit object on the page with the name `UserName`.

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

The following description, however, refers to the second item in the list above, as that is the first object of any type (**WebElement**) with the name `UserName`.

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

Navigating to a Line in the Expert View

You can use the Go To dialog box to navigate to a line in the Expert View.

To navigate to a line in the Expert View:

- 1** Click the Expert View tab.
- 2** Choose **Edit > Go To**. The Go To dialog box opens.
- 3** Enter the line of script to which you want to move in the **Line Number** box. The cursor moves to the line of script you indicated.

Enhancing Tests with Comments, Calculations, and Control-Flow Statements

The Virtual User Recorder 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 LoadTest 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 4, “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, the Virtual User Recorder does not process comments. Use comments to explain sections of a test script in order to improve readability and to make tests easier to update. Comments are displayed in green. For example:

```
'Sets the word "mercury" into the "password" edit field.
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").
    Set "mercury"
```

Note: You can also add a comment line using VBScript's **Rem** method. For additional information, refer to the *VBScript Reference* (choose **Help > Microsoft 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 method

```
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 LoadTest to execute one or more statements a specified number of times. It has the following syntax:

```
for counter = start to end [Step step]  
    statement
```

Next

counter the variable used as a counter.

start the start number of the counter.

end the last number of the counter.

step the number to increment at the end of each loop. The default is 1.

statement the statement to be executed during the loop.

In the following example, the Virtual User Recorder 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
```

“For...Each” Statement

A For...Each loop instructs the Virtual User Recorder to execute one or more statements for each element in an array or an object collection. It has the following syntax:

```
For Each item In array
    statement
Next
```

<i>item</i>	a variable representing the element in the array.
<i>array</i>	the name of the array.
<i>statement</i>	a statement or series of statements to be executed during the loop.

“Do...Loop” Statement

The Do...Loop statement instructs the Virtual User Recorder 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
```

<i>condition</i>	a condition to be fulfilled.
<i>statement</i>	a statement or series of statements to be executed during the loop.

The Virtual User Recorder 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
```

“While” Statement

A **While** statement instructs the Virtual User Recorder 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, the Virtual User Recorder performs a loop using the **While** statement while the number of passengers is fewer than four. Within each loop, the Virtual User Recorder increments the number by one.

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").QueryValue("value")
while passengers < 4
    passengers = passengers + 1
wend
```

“If...Then...Else” Statement

The **If...Then...Else** statement instructs the Virtual User Recorder 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, the Virtual User Recorder 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

```

“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")
```

“With” Statement

The **With** statement enables you to omit the object from a block of lines (from the **With** statement until the **End With** statement), so that lines beginning with “.” are treated as though the object is written before the “.” It has the following syntax:

```
With object
    statements
End With
```

object an object or a function that returns an object.
statements one or more statements to be executed on an object.

The **With** statement is a “shortcut” in VBScript. For example, you can write:

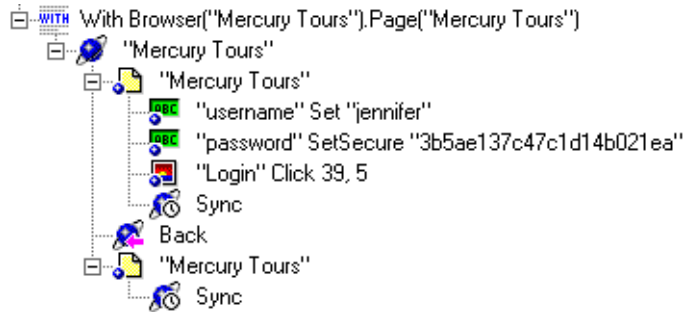
```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username")
    .Set "jennifer"
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("password")
    .SetSecure "3b5ae137c47c1d14b021ea"
Browser("Mercury Tours").Page("Mercury Tours").Image("Login").Click 39,5
Browser("Mercury Tours").Page("Welcome to Mercury").Sync
Browser("Mercury Tours").Back
Browser("Mercury Tours").Page("Mercury Tours").Sync
```

more concisely as:

```
With Browser("Mercury Tours").Page("Mercury Tours")
    .WebEdit("username").Set "jennifer"
    .WebEdit("password").SetSecure "3b5ae137c47c1d14b021ea"
    .Image("Login").Click 39,5
    .Sync
Browser("Mercury Tours").Back
```


.Sync
End With

This is displayed as follows in the Tree View:



Note that each line following the **With** statement is displayed as if it were written out fully in the Expert View, with the object name at the beginning of each line.

Note: Astra LoadTest includes Microsoft's *VBScript Language Reference*. The VBScript Language reference describes VBScript in detail. To open this reference, choose **Help > VBScript Reference**.

Retrieving and Setting Test Object Property Values

Test object properties are the set of properties defined by The Virtual User Recorder for each object. You can set and retrieve a test object's property values, and you can retrieve the values of test object properties from a runtime object.

When you run your test The Virtual User Recorder creates a live instance of the test object that is stored in the test object repository. You use the **GetProperty** and **SetTOProperty** methods to set and retrieve the test object property values of the *test object*.

The **GetProperty** method enables you to retrieve a property value or all the properties and values that The Virtual User Recorder uses to identify an object.

The **SetTOProperty** method enables you to modify a property value that The Virtual User Recorder uses to identify an object.

Note: Because The Virtual User Recorder refers to the live instance of the test object during the test run, any changes you make using the **SetTOProperty** method apply only during the course of the test run, and do not affect the values stored in the test object repository.

For example, the following statements would set the Submit button's name value to "my button", and then retrieve the value "my button" to the ButtonName variable:

```
Browser("QA Home Page").Page("QA Home  
Page").WebButton("Submit").SetTOProperty("Name", "my button")  
ButtonName=Browser("QA Home Page").Page("QA Home  
Page").WebButton("Submit").GetProperty("Name")  
Y
```

You use the **QueryValue** method to retrieve the current value of a test object property from a *run-time object in your application*.

For example, you can retrieve the target value of a link during the test run as follows:

```
Browser("Mercury Technologies").Page("Mercury  
Technologies").Link("Jobs").QueryValue("href")
```

For a list and description of test object properties supported by each object, refer to the *Astra LoadTest Function Reference*.

Accessing Run-time Object Properties and Methods

If the test object methods and properties do not provide the functionality you need, you can access the internal (or DOM) methods and properties of any run-time object using the **Object** property. You can also use the attribute Web object property to identify Web objects in your application according user-defined properties.

Tip: If you do not know the properties and methods of objects in your Web site or application, you can view them using the Object Spy. For information on the Object Spy, see Chapter 2, “Understanding the Test Object Model.”

Retrieving Run-time Object Properties

You can use the **Object** property to access the internal properties of any run-time object. For example, you can retrieve the current value of the ActiveX calendar’s internal **Day** property as follows:

```
Dim MyDay
Set MyDay=
Browser("index").Page("Untitled").ActiveX("MSCAL.Calendar.7").Object.Day
```

For additional information about the **Object** property, refer to the *Astra LoadTest Function Reference*.

Activating Run-time Object Methods

You can use the **Object** property to activate the internal methods of any run-time object. For example, you can activate the edit box’s internal **focus** method as follows:

```
Dim MyWebEdit
Set MyWebEdit=Browser("Mercury Tours").Page("Mercury
Tours").WebEdit("username").Object
MyWebEdit.focus
```

For additional information about the **Object** property, refer to the *Astra LoadTest Function Reference*.

Using Object Properties

The Internet Explorer Document Object Model (DOM) is a set of COM objects that correspond to the elements you see on a Web page. Within Internet Explorer, every HTML element (, <A>, <TABLE>, and so on) is programmable via an object that is part of the overall object model.

You can modify the appearance and behavior of an HTML element by altering an object's properties and calling its methods. In addition to methods and properties, the objects also fire events to signal user interaction or changes in the corresponding HTML element.

To allow access to these objects, Internet Explorer creates a top-level document object for each HTML document it displays. This document object represents the entire page. From this document object you can access the rest of the object hierarchy by using properties and collections. For example, you will use the *links* property of the document object to retrieve the actual link collection.

When you use the Object property of a Web Element in your script, you actually get a reference to the DOM object. This means that every action you can perform on the DOM object, you can also perform on the Web Element with the Object property.

Example 1

```
document.location.href =www.mercuryinteractive.com
```

will have the same functionality as:

```
Browser(browser_name).page(page_name).Object.location.href =  
www.mercuryinteractive.com
```

because Browser(browser_name).page(page_name).Object is the DOM's document.

Example 2

If you have an ActiveX Control embedded in the HTML file, you can access its properties and activate its methods using the Object Property.

```
Browser("index").Page("Untitled").ActiveX("MSCAL.Calendar.7").Object.Day =
20
```

In this example, `Browser("index").Page("Untitled").ActiveX("MSCAL.Calendar.7").Object` is a reference to the calendar ActiveX itself, and `Day` is its property. You can Set this property (like in the example), or Get it.

Example 3

Suppose you have a the following statement in your script:

```
document.MyForm.MyHiddenField.value = My New Text
```

The following example achieves the same thing by using the `Object` property, where `MyDoc` is the DOM's document:

```
Dim MyDoc
Set MyDoc = Browser(browser_name).page(page_name).Object
MyDoc.MyForm.MyHiddenField.value = My New Text
```

Example 4

In this example, `LinksCollecton` is assigned to the link collection of the page through the `Object` property. Then, a message box pops for each of the links, with its inner text.

```
Dim LinksCollection, link
Set LinksCollection = Browser(browser_name).Page(page_name).Object.links
For Each link in LinksCollection
    MsgBox link.innerHTML
Next
```

Additional Information

For more information about the DHTML Document Object Model, refer to:

<http://msdn.microsoft.com>

21

Working with Astra LoadTest—for Power Users

This chapter answers some of the questions that are asked most frequently by *advanced users* of Astra LoadTest. The questions and answers are divided into the following sections:

- ▶ Recording and Running Tests
- ▶ Working with Dynamic Web Content
- ▶ Advanced Web Issues
- ▶ Test Maintenance
- ▶ Testing Localized Applications
- ▶ Load Testing Questions

Recording and Running Tests

- ▶ **How does the Virtual User Recorder capture user processes?**

The Virtual User Recorder hooks the browser (Netscape, Microsoft Internet Explorer, or AOL). As the user navigates the Web site, the Virtual User Recorder intercepts and records all steps as they enter the browser. The Virtual User Recorder can then run the test by running the steps as they originally occurred.

- ▶ **How does the Virtual User Recorder record and identify objects on Web pages?**

The Virtual User Recorder can record all Web objects on a Web page. Each HTML tag is considered a Web object. The Virtual User Recorder identifies

each object by its HTML tag and logical name and stores the descriptions of each object in the memory.

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 LoadTest will recognize them when it runs the test.

- ▶ **How can I check that a spawned window exists (or does not exist)?**

Sometimes a link in one window spawns another window. Use the **Exist** method to check whether or not a spawned window exists. For example:

```
Browser("Window_logical_name").Exist
```

For additional information about the **Exist** method, refer to the *Astra LoadTest Function Reference*.

- ▶ **How does the Virtual User Recorder record on dynamically generated URLs and Web pages?**

The Virtual User Recorder actually clicks on links as they are displayed on the page. Therefore, the Virtual User Recorder 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 the Virtual User Recorder records the “IMG” HTML tag, and the name of the image. This enables Astra LoadTest to find this image in the future and click on it.

For additional information, see “Understanding Dynamic Descriptions of Objects” on page 41.

- ▶ **How can I extract data from HTTP responses and use that data as dynamic parameters?**

Astra LoadTest has the ability to extract any data (header information, HTML information, Javascript, etc.) from HTTP responses. Once the data has been extracted, it can be retrieved and used as parameter data.

The methods you use to perform these actions are the WebUtil methods **SaveResponseData** and **GetSavedResponseData**. These methods are only available when using TurboLoad technology.

You use the **SaveResponseData** method before a navigation. This registers a request for a parameter search in the response to the navigation. You can specify any number of SaveResponseData requests before a given navigation.

You use the **GetSavedResponseData** to retrieve the extracted value. Once the value is retrieved it can be used as a parameter.

For additional information about the **SaveResponseData** and **GetSavedResponseData** methods, refer to the *Astra LoadTest Function Reference*.

Advanced Web Issues

► How does Astra LoadTest 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 LoadTest stores cookies in the memory for each Vuser, and the browser handles them as it normally would. In Astra LoadTest, a log is made every time a cookie is set or accessed.

► How does Astra LoadTest 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 LoadTest.

► How does Astra LoadTest 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 LoadTest or the browser.

► How does Astra LoadTest 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 LoadTest has no problem handling meta tags.

► Does Astra LoadTest 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 LoadTest.

► **Does Astra LoadTest work with COM?**

Astra LoadTest complies with the COM standard.

Astra LoadTest supports COM objects embedded in Web pages (which are currently accessible only using Microsoft Internet Explorer).

► **Does Astra LoadTest 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 LoadTest supports XML and recognizes XML tags as objects.

Test Maintenance

► **How do I maintain my test when my application changes?**

The way to maintain a test when your application changes depends on how much your application changes. This is one of the main reasons you should create a small group of tests rather than one large test for your entire application. When your application changes, you can rerecord part of a test. If the change is not significant, you can manually edit a test to update it.

You can also use Astra LoadTest'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 12, "Working with Actions."

Testing Localized Applications

- ▶(I am testing localized versions of a single application, each with localized user interface strings. How do I create efficient tests in Astra LoadTest?

You can parameterize these user interface strings using parameters from the global Environment variable list. This is a list of variables and corresponding values that can be accessed from any test. For additional information, see Chapter 7, “Parameterizing Tests.”

- ▶(I am testing localized versions of a single application. How can I efficiently input different data in my tests, depending on the language of the application?

You can create an external data table for each localized version of your application. When you change the localized version of the application you are testing, you simply switch the data table for your test. For additional information, see Chapter 13, “Working with Data Tables.”

- ▶(I am testing localized versions of a single application. How can I efficiently instruct The Virtual User Recorder to handle exceptions, when the exception strings vary, depending on the language of the application?

The Virtual User Recorder stores information about handling unexpected events and errors in an external exception configuration file. This file, *Exception.inf*, is located in the *[Astra LoadTest installation]\dat* folder. When you change the localized version of the application you are testing, you simply switch the *Exception.inf* exception configuration file. For additional information, see Chapter 14, “Handling Unexpected Events and Errors.”

Load Testing Questions

- ▶ How does Astra LoadTest handle think times?

Think times can be added to the script at any place; they can also be adjusted. Generally random think times are between 2 and 6 seconds, this margin can be changed to whatever you like.

► **(Can I use the same scripts for functional/regression, load testing and monitoring? Do I need to re-record them or edit the scripts?)**

Yes and no. It depends highly on how you create the scripts. It is possible to write one script that does everything, though it generally isn't recommended. We recommend that you create smaller scripts that have a specific functionality for a specific area of your Web site, then link all of these together in larger, more dynamic scripts. This allows you to do a wide variety of testing. The current version of Astra LoadTest has added functionality that makes modular scripts more viable than previously. Also, Astra QuickTest scripts can be used in LoadTest, so you could use them later for load testing if you determined that they were a good representation of your user actions. It's very much up to you what you do with the scripts.

► **(At what point do I consider using LoadRunner over the Astra LoadTest tools? Can I integrate the scripts between products?)**

Astra LoadTest scripts can be used in LoadRunner. LoadRunner should probably be used if you have a site that uses Java or other complex components. LoadRunner can run more Vusers and can run on the HTTP only level, whereas Astra LoadTest runs over a browser layer and thus is more dynamic. Astra LoadTest doesn't offer quite as much functionality as LoadRunner. You can perform database load testing with LoadRunner, whereas Astra LoadTest is just Web based. LoadRunner is generally for larger enterprise applications.

► **(Availability testing: Is there a monitoring tool integrated with Astra tools?)**

Astra LoadTest has numerous graphs and monitors that allow you to watch how the server is doing under load.

► **(Do JavaScript or session management require me to create scripts in more than one phase. (Record the initial interaction and then program the script to make it playback correctly with session management)?)**

No. We use an internal browser that handles all of these complexities. You just record the script, and play it back in the Virtual User Recorder or the Controller.

Part V

Configuring the Virtual User Recorder

22

Setting the Virtual User Recorder Testing Options

You can control how the Virtual User Recorder records and runs tests by setting testing options.

This chapter describes:

- Setting the Virtual User Recorder Testing Options
- Selecting The Virtual User Recorder Testing Options

About Setting the Virtual User Recorder Options

The Virtual User Recorder testing options affect how you record and run tests. For example, you can set the speed at which The Virtual Recorder runs a test, or set the timing-related settings used by The Virtual Recorder. The values you set remain in effect for all tests and for subsequent testing sessions.

You can also set testing options that affect only the test currently open in The Virtual User Recorder. For more information, see Chapter 23, “Setting Testing Options for a Single Test.”

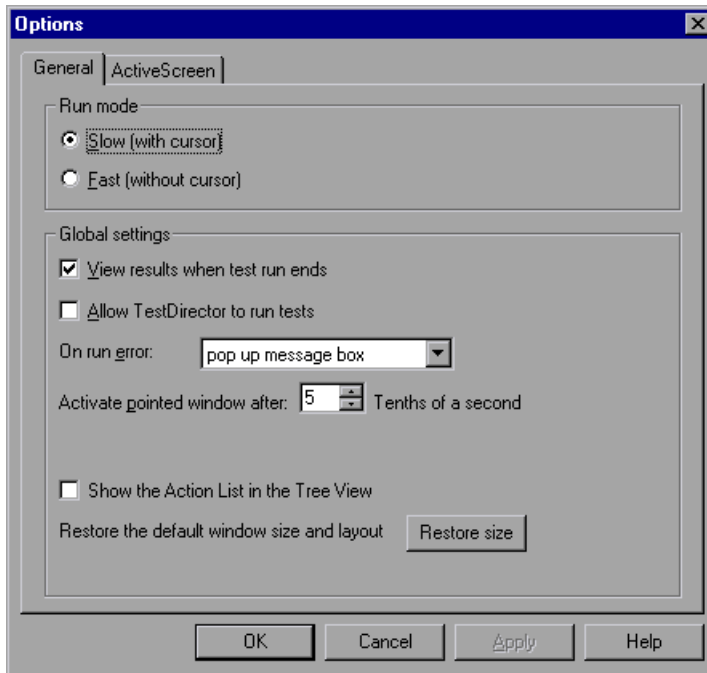
Setting the Virtual User Recorder Testing Options

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 the Virtual User Recorder testing options:

- 1 Choose **Tools > Options**.

The Options dialog box opens. It is divided by subject into two tabbed pages.



- 2 To choose a page, click a tab.
- 3 Set an option, as described in “Selecting The Virtual User Recorder Testing Options”.
- 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.

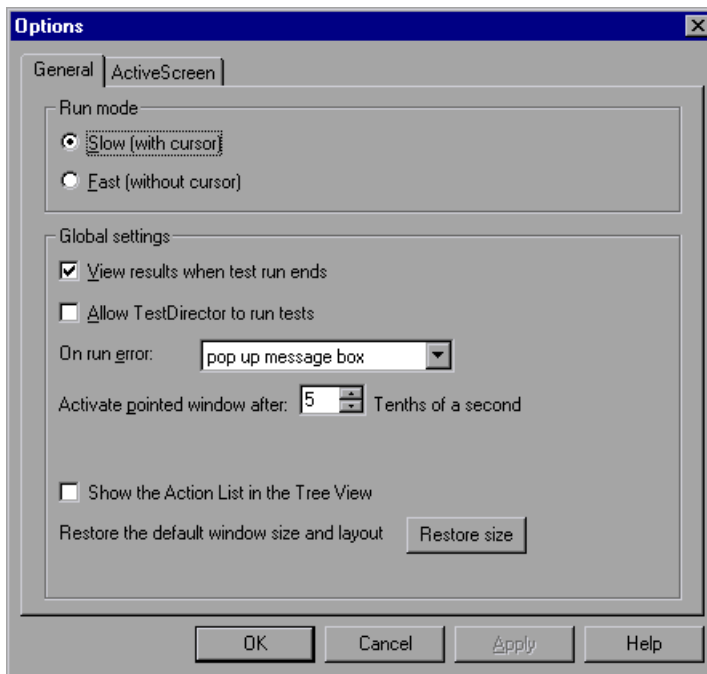
Selecting The Virtual User Recorder Testing Options

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

General Testing Options

The General tab options affect how the Virtual User Recorder runs tests and displays test results.

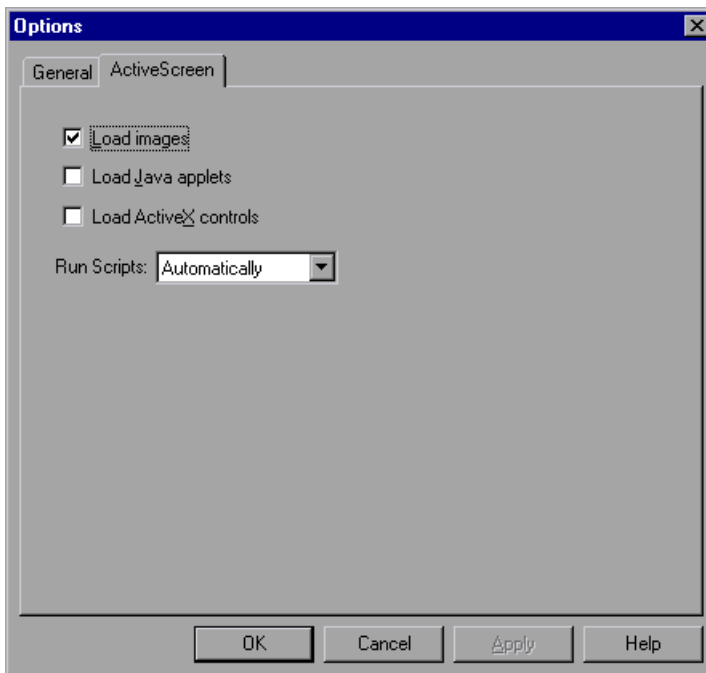


The General tab includes the following options:

Option	Description
Run mode - Slow (with cursor)	Instructs Astra LoadTest 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 Astra LoadTest <i>Installation Guide</i> .
Run mode - Fast (without cursor)	Instructs Astra LoadTest 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 LoadTest to display the test results automatically following the test run.
Allow TestDirector to run tests	Enables TestDirector to remotely run tests.
On run error	Determines how Astra LoadTest 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.
Activate pointed window after	Specifies the time (in tenths of a second) that Astra LoadTest waits before it sets focus on the Web browser.
Show the Action List in the Tree View	Determines whether the Action List is displayed in the Tree View. If your test contains reusable actions, this option is automatically enabled.
Restore the default window size and layout	Restores the Virtual User Recorder window so that it displays the default panes in the default sizes.

ActiveScreen Testing Options

The ActiveScreen tab options affect how the Virtual User Recorder displays Web pages in the ActiveScreen.



The ActiveScreen tab includes the following options:

Option	Description
Load images	Instructs Astra LoadTest to load images from your browser page to the ActiveScreen pane.
Load Java applets	Instructs Astra LoadTest 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.

Option	Description
Load ActiveX controls	<p>Instructs Astra LoadTest 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.</p>
Run scripts	<p>Determines how Astra LoadTest runs scripts during a test run.</p> <p>Choose an option from the list:</p> <p>Always instructs Astra LoadTest to run scripts when loading a page in the ActiveScreen.</p> <p>Automatically allows Astra LoadTest to determine if scripts need to be run, and to run them when needed.</p> <p>Never instructs Astra LoadTest not to run scripts when loading a page in the ActiveScreen.</p>

23

Setting Testing Options for a Single Test

You can control how Astra LoadTest records and runs specific tests by setting testing options.

This chapter describes:

- ▶ Setting Testing Options for a Single Test
- ▶ Testing Options for a Single Test
- ▶ Tuning the Test Replay

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 LoadTest to run a parameterized action for only certain lines in the table in the Data pane. You can also teach Astra LoadTest to recognize a specific object in your test as a standard object. These testing options are saved when you save the test.

You can set testing options from within a test, for a part of the test, using a test script. For more information, see Chapter 25, “Setting Testing Options from a Test Script.”

You can also set testing options that affect all tests. For more information, see Chapter 22, “Setting the Virtual User Recorder Testing Options.”

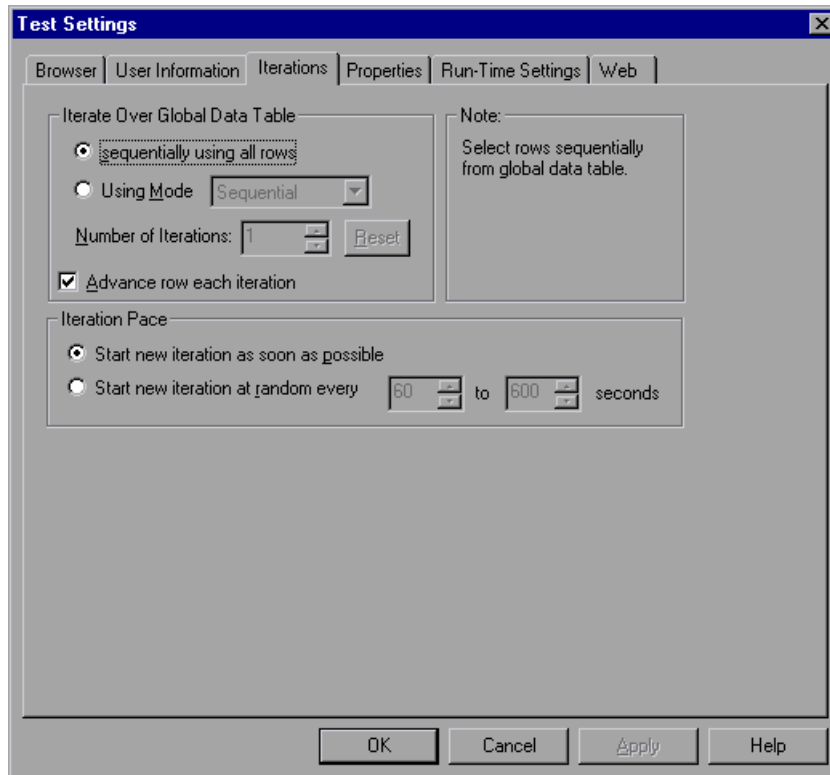
Setting Testing Options for a Single Test

Before you record or run a test, you can use the Test Settings dialog box to modify your testing options. Some of the settings affect the recording of your test in the Virtual User Recorder, while others affect the replay under load in the Controller.

To set testing options for a single test:

- 1 Choose **Test > Settings**.

The **Test Settings** dialog box opens. It is divided by subject into six tabbed pages.



- 2 To choose a page, click a tab.
- 3 Set an option, as described in “Testing Options for a Single Test”.

- 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.

Note: The Run-Time tab is described in a separate section of the manual. For information on these settings, see Chapter 9, “Testing Load.”

Testing Options for a Single Test

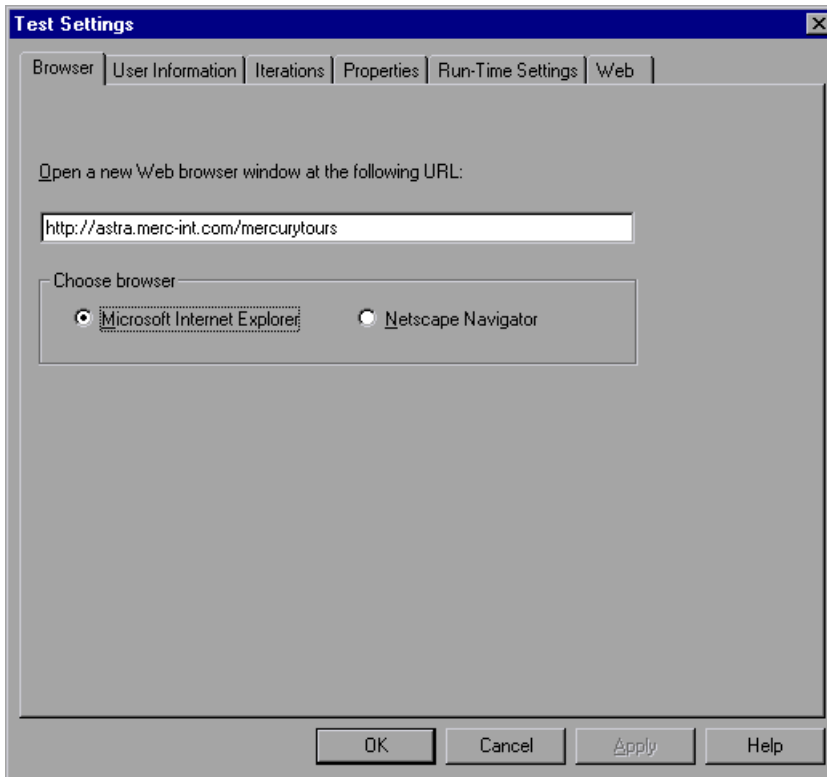
The Test Settings dialog box contains the following tabbed pages:

Tab Heading	Subject
Browser	Options for setting a Web browser for recording tests
User Information	Options to handle network and active screen passwords
Iterations	Options for setting test run logic for tests and actions
Properties	Options for setting the properties of tests
Run-Time Settings	Options for defining how your test runs
Web	Options for recording tests

This section lists the testing options you can set using the Test Settings dialog box.

Browser Testing Options

The StartUp tab options set which browser to use while recording and whether to use an existing Web browser window or to open a new browser window to a specified location.



The Browser tab includes the following options:

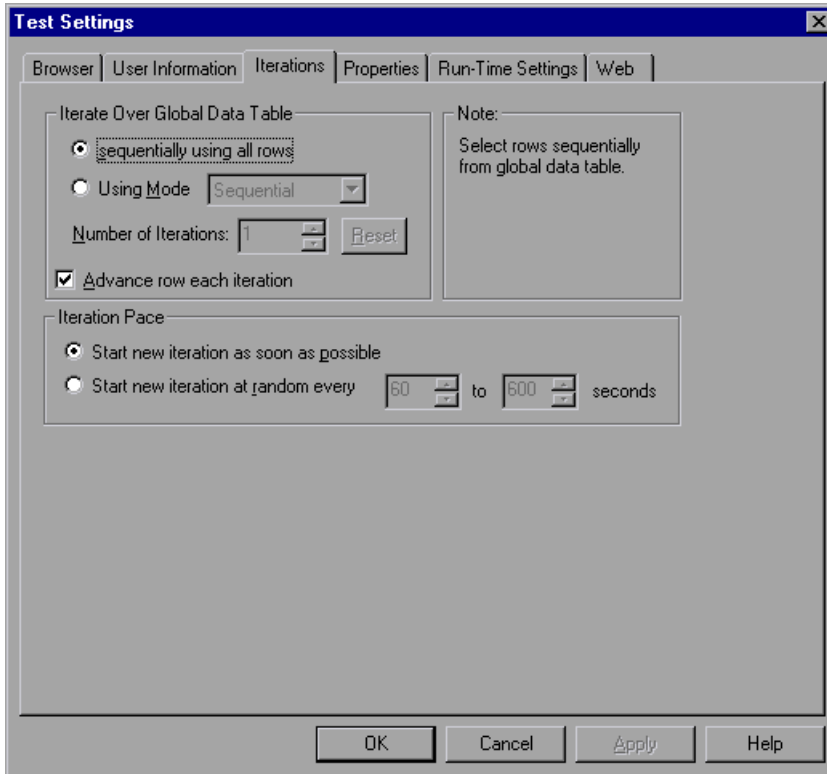
Option	Description
Open new Web browser window at the following URL	Instructs Astra LoadTest to open a new browser session to record a test using the specified Web location address.
Choose browser	Instructs Astra LoadTest to use the specified browser type to record a browser session.

Note: You can also set the **Use existing Web browser window**, **Open new Web browser window at the following URL** and **Choose browser** options for a specific test in the **Start Recording** dialog box, which opens when you start recording a new test.

Iterations Testing Options

When you run a test, Astra LoadTest performs the steps you recorded on your Web site. When you run a test with global parameters, Astra LoadTest

runs the test for each row in the table in the Data pane, using the parameters you specified.

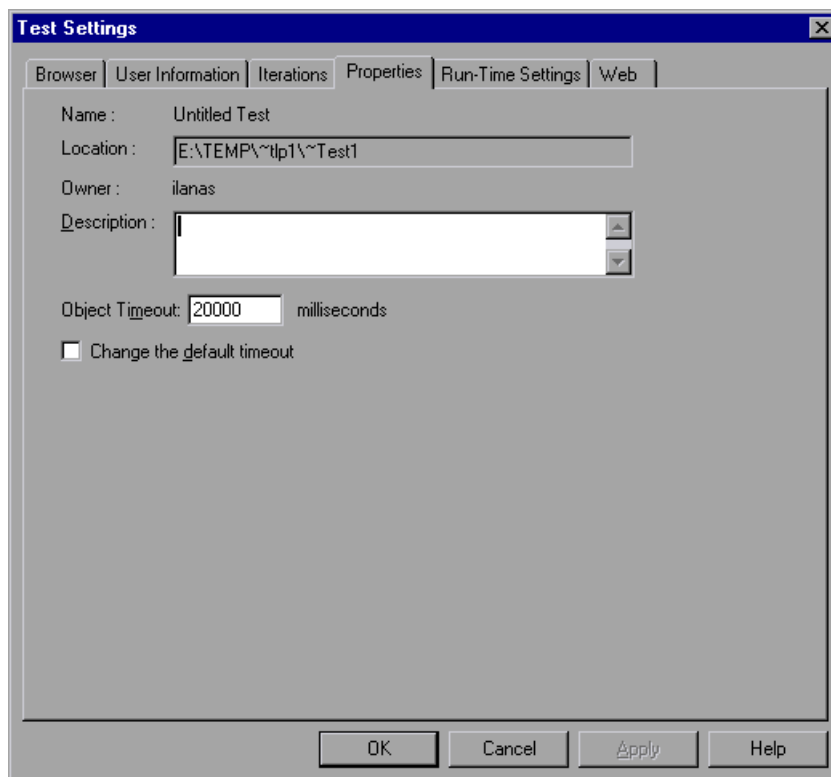


The Iterations tab includes the following options:

Option	Description
Iterate Over Global Data Table	Runs the test or action (depending on which is highlighted) using all the values in the local data table (for an action) or in the global data table (for a test). Use Sequentially using all rows to run as many iterations as there are rows in the data table. Alternatively, you can specify the Number of Iterations , regardless of the number of rows in the data table.
Using Mode	Specifies how Astra LoadTest should assign data to the Users for each iteration. Sequential assigns the data values to a User in a sequential order. Random assigns the data values in a random order. Unique indicates that Astra LoadTest should make sure that each User uses a unique data value for each iteration. The mode setting is only relevant when running the scenario in the Controller. While running the test in the Virtual User Recorder, parameters are taken from the Data table in the order in which they appear.
Advance row each iteration	Specifies whether to use a new row of values from a data table for each global iteration. By default, the option is selected. This instructs the User to select the next row of data for each iteration.
Iteration Pace	Specifies how long the test should wait before continuing with the next iteration.

Properties Testing Options

The Properties tab option defines general test information.



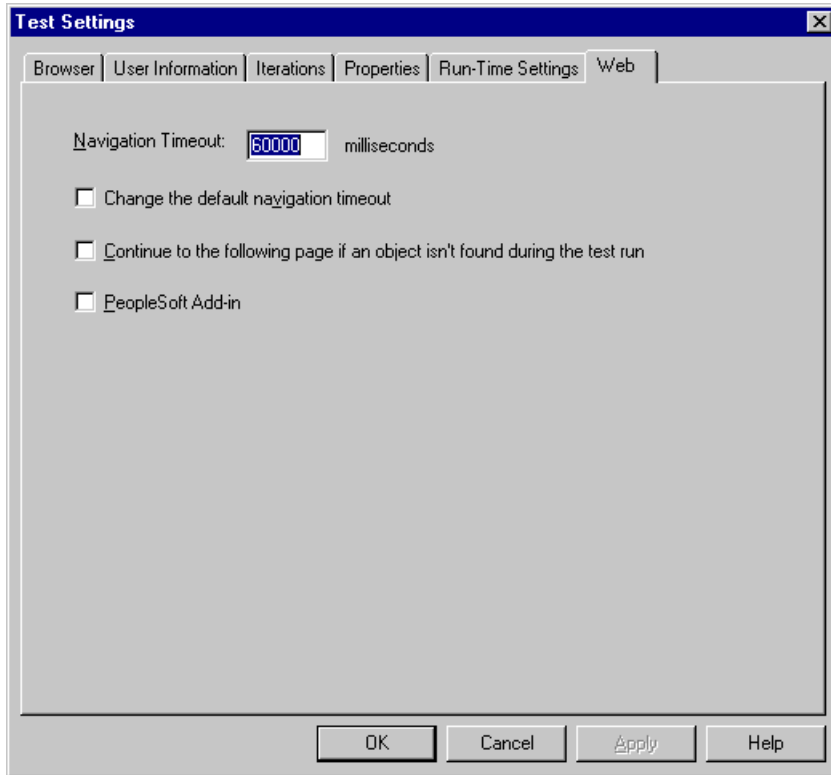
The Properties tab includes the following options:

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.

Option	Description
Object Timeout	Instructs Astra LoadTest not to exceed the specified time while loading an object.
Change the default timeout	Instructs Astra LoadTest not to use the default timeout.

Web Testing Options

The Web tab options affect the recording in the Virtual User Recorder.



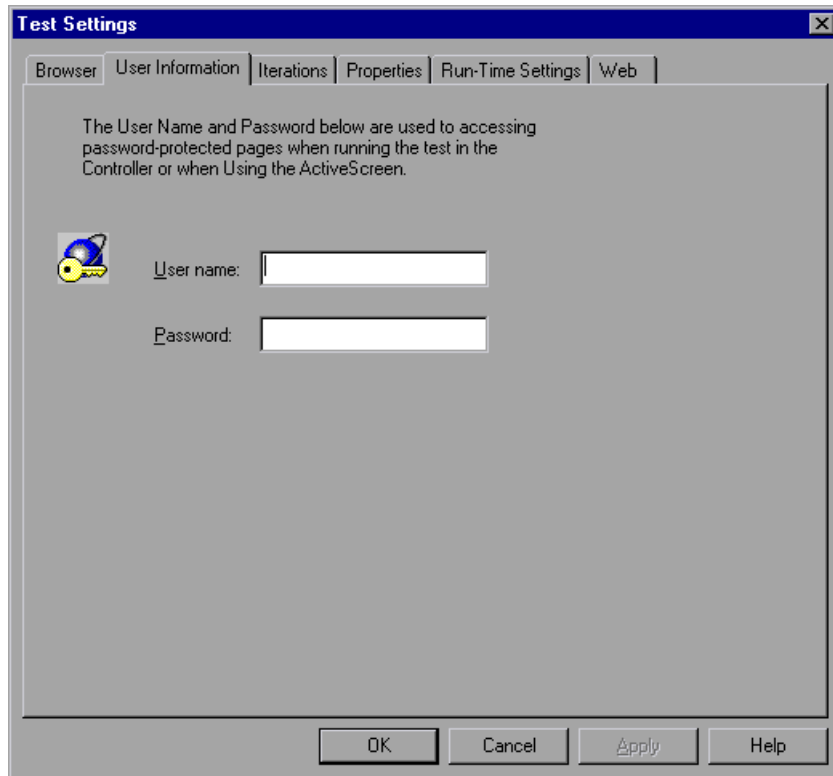
The Web tab includes the following options:

Option	Description
Navigation timeout	Indicates the interval (in seconds) Astra LoadTest 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	Specifies that Astra LoadTest should continue running the test, even if an object is not found. (See Alternative Navigation Properties below.)
PeopleSoft Add-in	Enable this option to replay a test that navigates to a PeopleSoft server.

User Information Settings (for NTLM authentication)

The User Information tab allows you to enter network or active screen passwords for processing tests in applications where passwords are required.

At the start of a recording session, there are no values in either the User name or Password fields. The first time you navigate to a page that requires this information, it will be stored for use during the running of the test.



If authentication is required during a test run, the recorded values are used to supply the user name and password for all Users.

The Virtual User Recorder only stores the first set of data recorded and uses it throughout your test. In some tests however, there may be a need to supply multiple passwords. Perhaps you are going to run several Users, each requiring a set of unique data. Alternatively, there may be a resource requiring a separate set of data.

To supply multiple passwords in a test, use the **webutil.SetAuthenticationPassword** or **webutil.SetAuthenticationUsername** functions in the Expert view. These methods need to be added to the script

before the step in the test which will utilize them. There is no limit as to the number of times they can appear in a script. For more details see the *Astra LoadTest Function Reference*.

If the recording takes place where no authentication is recorded because you are within a domain, then no values are added to the User Information tab. If, however, your Web site requires authentication, the lack of recorded values may keep the test from running the Controller. If this problem occurs, you can add the information to the User Information tab manually or you can use the webutil functions.

The User Information tab includes the following options:

Option	Description
User Name	The Network or ActiveScreen user name.
Password	The Network or ActiveScreen password.

Tuning the Test Replay

Suppose you are running your test and an error such as “Object Not Found” occurs. When the error is encountered, the test run stops and Astra LoadTest issues an error message. You can resolve the error and run the test again.

In some cases it is not important to resolve an particular error. The error may have no effect on your test so you want the test to run as if the error does not exist. Use either the Alternative Navigation Properties or Replace with Alternative Navigation Properties options to bypass a problem on a particular page. See “Defining Alternative Navigation Properties” below.

If the error is related to a page timeout, and the test runs successfully after you bypass the page, use the Add Sync on Request option to synchronize the current page and navigate to the next page. See “Adding Test Synchronization” below.

In some cases an error occurs in load mode because information for a JavaScript request is not found. Use the Save Layout option to record the needed page information. See Using Save Layout below.

Defining Alternative Navigation Properties

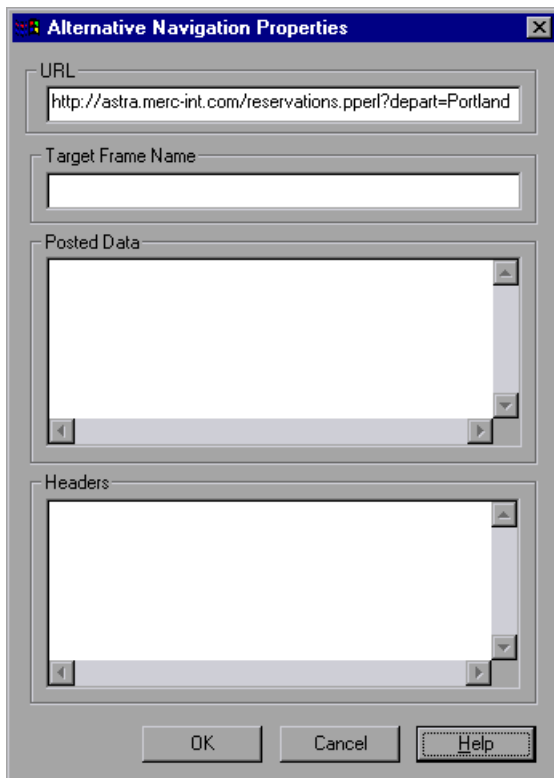
If a navigation step in a page fails during a test run and you have selected the **Continue to the following page if an object isn't found during the test run** option in the Web tab, Astra LoadTest Controller can use the alternative navigation properties as an alternate way to navigate to the next page in the test. You can set the alternative navigation properties for each page in your test.

It is important to note that before using the alternative navigation properties, an error must occur. If for example, the error is caused by a specific page not downloading, the Virtual User Recorder only uses the alternative navigation properties after the navigation time out is reached.

To set alternative navigation properties:

- 1 Right-click on a page or step icon in the test tree view or on a script line in the Expert view and select **Alternative Navigation Properties**. Alternatively, select the icon or script line and choose **Step > Alternative Navigation Properties**.

The Alternative Navigation Properties dialog box opens.



2 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.

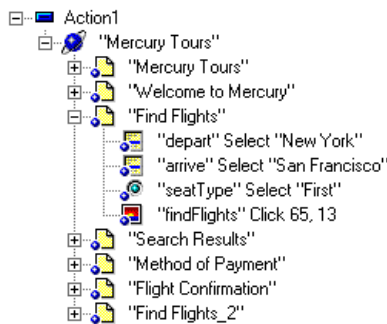
- Repeat steps 1 - 2 for each page for which you want to set alternative navigation properties.

Replacing with Alternative Navigation Properties

Using the Replace with Alternative Navigation Properties option provides additional flexibility while running your test.

By specifying this option, your test proceeds to the next URL without waiting for an error to occur. For example, suppose a particular page does not load properly and causes a time out, by proceeding directly to the specified URL, your test is not affected by the load time of the problematic page. This allows you to run your test with no errors and without the delays incurred by using the alternative navigation properties.

For example, suppose you have recorded the following test:



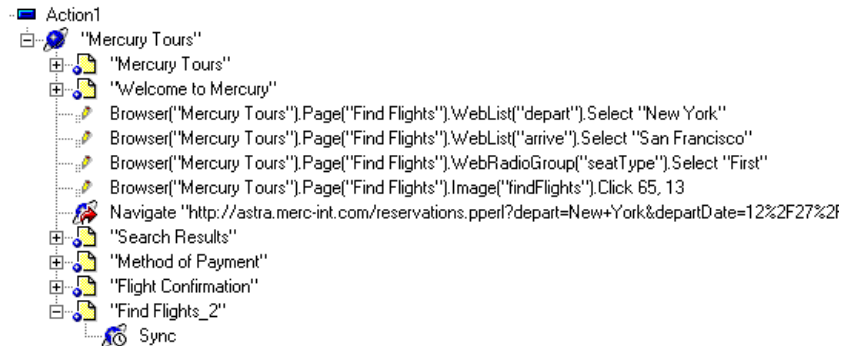
When you run the test, you find that the “Find Flights” page is causing an error. You decide that the error is not something you want to resolve immediately. Instead you want the test to navigate directly to the “Search Results” page from the “Welcome to Mercury” page, skipping the problematic page. You do this by using the Replace with Alternative Navigation Properties option.

To specify Replace with Alternative Navigation Properties:

- Right-click on a page or step icon in the test tree view and select **Replay Tuning Steps > Replace with Alternative Navigation Properties**. Alternatively, select the icon or script line and choose **Insert > Replay Tuning Steps > Replace with Alternative Navigation Properties**. You are prompted to continue the process.

2 Click Yes.

The Virtual User Recorder inserts a navigation line and changes the recorded steps of the “Find Flights” page into comments.



The following is the same test in the Expert View.

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set "mercury"
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("password").SetSecure "3a485c43f5cd0272f"
Browser("Mercury Tours").Page("Mercury Tours").Image("Login").Click 35, 11
Browser("Mercury Tours").Page("Welcome to Mercury").Check CheckPoint("Welcome to Mercury")
Browser("Mercury Tours").Page("Welcome to Mercury").Image("Dearch Flights").Click
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select "New York"
Browser("Mercury Tours").Page("Find Flights").WebList("arrive").Select "San Francisco"
Browser("Mercury Tours").Page("Find Flights").WebRadioGroup("seatType").Select "First"
Browser("Mercury Tours").Page("Find Flights").Image("findFlights").Click 65, 13
Browser("Mercury Tours").Navigate "http://astra.merc-int.com/reservations.pperl?depart=New+York&departDate=12%2F27%2f"
Browser("Mercury Tours").Page("Search Results").Image("reserveFlights").Click 88, 16
Browser("Mercury Tours").Page("Method of Payment").Image("buyFlights").Click 95, 10
Browser("Mercury Tours").Page("Flight Confirmation").Image("Book Another").Click 14, 12
Browser("Mercury Tours").Page("Find Flights_2").Sync
```

The lines that appear in italics are the commented lines and the navigation line appears directly after them.

Parameterizing the Replace with Alternative Navigation Properties

Using the Replace with Alternative Navigation Properties option also allows you to parameterize the parameters of the navigate method.

In the examples above you can see that navigation line contains all of the data of the alternative navigation properties. Any field can be parameterized.

To parameterize the parameters of the navigate method:

- 1** In the tree view, right-click the navigation line and select **Function Arguments**. The Function Arguments dialog box opens.
- 2** In the Edit value section select the **Parameter** radio button.
- 3** Enter a meaningful name for the parameter in the name box.
- 4** Choose whether the parameter is global or local.
- 5** Click **OK**. The name of your parameter is entered in the data pane and the navigation line is entered as the value in the first row.

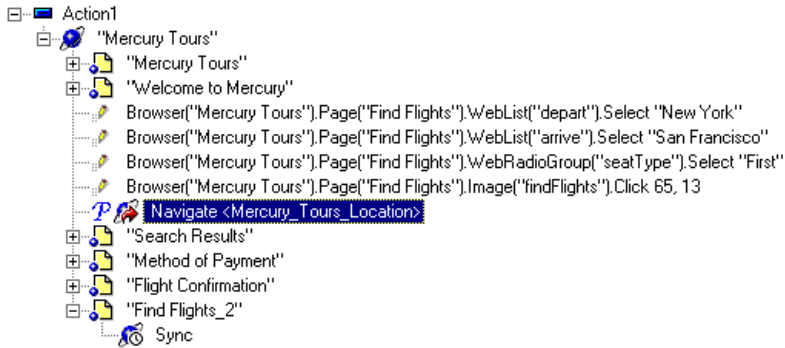
	Mercury	Tours	Location	B	C	D	E	F	G	H
1	http://astra.merc-int.com/reservations.pperl?depart=New+York&departDate=12%2F27%2F2000&arrive									
2										
3										
4										
5										
6										
7										

- 6** Replace the data you want to parameterize in the displayed line. To add additional rows of data, copy and paste the first row, replacing the data you want to parameterize in each row.

If you want to parameterize the value New + York and enter two additional cities the data pane will be similar to the following:

	Mercury	Tours	Location	B	C	D	E	F	G	H
1	http://astra.merc-int.com/reservations.pperl?depart=New+York&departDate=12%2F27%2F2000&arrive=San+Franci									
2	http://astra.merc-int.com/reservations.pperl?depart=San+Diego&departDate=12%2F27%2F2000&arrive=San+Franci									
3	http://astra.merc-int.com/reservations.pperl?depart=San+Juan&departDate=12%2F27%2F2000&arrive=San+Francis									
4										
5										
6										
7										

Your tree view will replace the navigation line string with the parameter name and will add the parameter symbol to the line.



Note: This solution may require correlation. You can obtain the correlation information by using the *weutil SaveResponseData* method with the TurboLoad mode enabled.

Returning your Test to the Original Format

Once you no longer want to navigate past the problematic page, you will want to change your test back to its original form.

To undo the Replace with Alternative Navigation Properties option:

- 1 In the Expert View, delete the leading apostrophe of the comment lines and they will revert to active lines.
- 2 Delete the entire navigation line.

Your test is now the same as the original recording.

Adding Test Synchronization

During a test run, Astra LoadTest waits for all elements on a page to load before continuing with the next step in the business process.

Astra LoadTest knows that the page is complete because it receives a completion notification from the browser. Upon receipt of the notification, all elements are loaded and the run should continue.

If the completion notification is not received before the Navigation time out, a message “The page has not finished downloading... xx HTTP Requests have been completed”, is issued and the test continues with the next step.

There are two reasons why the download time exceeds the navigation time out.

The first reason is that there is a Network problem and not all of the information downloaded before the navigation timeout.

Increasing the navigation time out will resolve this problem.

For more information on resolving page downloading problems, see the Page has not finished downloading help of the Troubleshooter.

The second reason is that the completion notification is never sent. This problem can occur for several reasons. For example, perhaps the page contains a flash or ActiveX item that is continually running, or maybe there is faulty JavaScripting (a function performs a document open(),document.write() without a corresponding document.close().)

In this case, instruct Astra LoadTest to continue the test run without receiving completion notification using the **Add Sync on Request**. Using this method, you indicate to Astra LoadTest to continue test execution even though some HTTP requests were not completed.

You specify the desired number of requests to complete, after which Astra LoadTest continues executing the test.

The recommended way to use this feature is to check the warning messages within the **Vuser Log** window. Locate the message: “The page has not finished downloading... xx HTTP Requests have been completed”. Use the number of completed requests (xx) as a value for the number of requests you want your test to complete before continuing.

Occasionally, the browser may send the completion notification before all the resources are downloaded. In this case, you can see that there are

additional HTTP requests being processed after the end of the automatic transaction, which is issued when the completion notification is received.

In these cases you can instruct Astra LoadTest to wait for the additional resources to be completed before it proceeds. Use the number of completed requests (xx) before the end of transaction *plus* the number of additional HTTP requests as a value for the number of requests you want your test to complete before continuing.

Astra LoadTest will ignore any completion notification that it receives until the required number of HTTP requests has been completed.

When you add synchronization to a page using the UI described below, Astra LoadTest adds a line to test containing the **WebUtil.SyncOnRequest** method and its arguments:

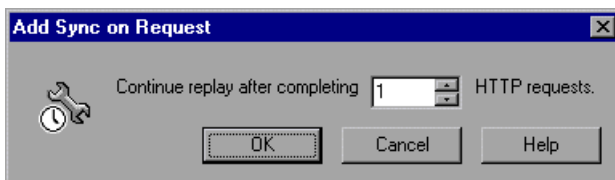
```
WebUtil.SyncOnRequest 4
```

For more information about this method, see the *Astra LoadTest Function Reference*.

To synchronize a Web page after a specified number of HTTP requests:

- 1 Select the problematic page in the test tree view—the page that comes after the warning in the Vuser Log. Perform a right-click and select **Replay Tuning Steps > Add Sync on Request**. Alternatively, select the icon or script line and choose **Insert > Replay Tuning Steps > Add Sync on Request**.

The Add Sync on Request dialog box opens:



- 2 Specify the number of HTTP requests to process before proceeding to the next operation. See above for information on choosing an appropriate value.

- 3 Click **OK** to insert the synchronization into your test. The Virtual User Recorder inserts a **WebUtil.SyncOnRequest** method before the selected step.

Using Save Layout

Every Web page is made up of various Web elements. When a page is displayed in the browser, the positioning of the Web elements on the page is determined dynamically by the browser.

When running a test under load, true browser pages are not used. This means that the Web element attributes are not available, and cause problems in certain JavaScripts. Consider the following section of script:

```
<SCRIPT>
function important ()
{
    elem1.style.width = elem2.offsetwidth - 200;
    document.write( 'A HREF= important.htm>Important Link </ A>' );
}
</SCRIPT>
<Table ID="elem1">...
<TR ID="elem2">...
```

The script runs fine in the browser, however, when running under load the “offsetwidth” property of elem2 is zero. This causes a negative value for the “width” property of elem1, resulting in a JavaScript exception and non-execution of the document.write statement.

The “Important” link will not be created on the page, which causes a replay failure if the link is referenced in the Vuser script.

The Save Layout option allows you to eliminate this problem. When you use the Save Layout option on a particular page, Astra LoadTest scans the page and reads the layout properties of all of its elements. The results are stored in the Object Repository and are retrieved as needed.

To use Save Layout for a particular page:

- 1 In the tree view, right-click a fully loaded page.

- 2** Select **Replay Tuning Steps > Save Layout**. A message appears asking you if you want to continue the Save Layout process.
- 3** Click **Yes**. A RestoreLayout step is added to the tree view and a corresponding line is added to the expert view.

When the test is run in load mode, the RestoreLayout will provide the information needed to run the JavaScript properly.

In the Expert View, RestoreLayout can be used with additional parameters to further enhance the page layout information. For more details, see the RestoreLayout method in the *Astra LoadTest Function Reference*.

24

Customizing the Expert View

You can customize the way your test is displayed when you work in the Expert View. The Virtual User Recorder 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 LoadTest.

This chapter describes:

- Setting Display Options
- Personalizing Editing Commands

About Customizing Your Test in the Expert View

The Virtual User Recorder'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 the Virtual User Recorder 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 LoadTest 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

The Virtual User Recorder 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

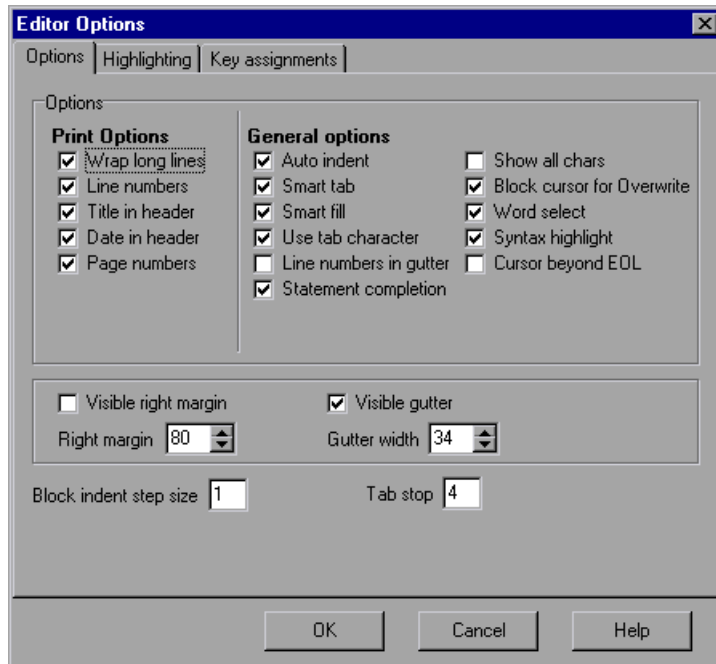
The Virtual User Recorder's display options let you control how test scripts appear in the Expert View tab, how different elements of test scripts are displayed, and how test scripts will appear when they are printed.

Customizing Test Scripts

You can customize how the Virtual User Recorder'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 Choose from the following options:

Options	Description
Auto indent	Causes lines following an indented line to automatically begin at the same point as the previous line. You can click the Home key on your keyboard to move the cursor back to the left margin.
Smart tab	A single press of the tab key will insert the appropriate number of tabs and spaces in order to align the cursor with the text in the line above.

Options	Description
Smart fill	Insert the appropriate number of tabs and spaces in order to apply the Auto indent option. When this option is not selected, only spaces are used to apply the Auto indent. (Both Auto indent and Use tab character must be selected to apply this option).
Use tab character	Inserts a tab character when the tab key on the keyboard is used. When this option is not enabled, the appropriate number of space characters will be inserted instead.
Line numbers in gutter	Displays a line number next to each line in the script. The line number is displayed in the test script window's gutter.
Statement completion	Opens a list box displaying all available matches to the method prefix whenever the user presses the Ctrl and Space keys simultaneously, the period (.) key, or chooses Edit > Complete Word . Select an item from the list to replace the typed string. To close the list box, press the Esc key. Displays a tooltip with the method parameters once the complete method name appears in the editor. The method parameters are displayed also whenever the user presses the Ctrl, Shift, and Space keys simultaneously or choose Edit > Parameter Info .
Show all chars	Displays all text symbols, such as tabs and paragraph symbols.
Block cursor for Overwrite	Displays a block cursor instead of the standard cursor when you select overwrite mode.
Word select	Selects the nearest word when you double-click in the Expert View tab.
Syntax highlight	Highlights test script elements such as comments, strings, or reserved words.
Cursor beyond EOL	Enables the Virtual User Recorder to display the cursor after the end of the current line.

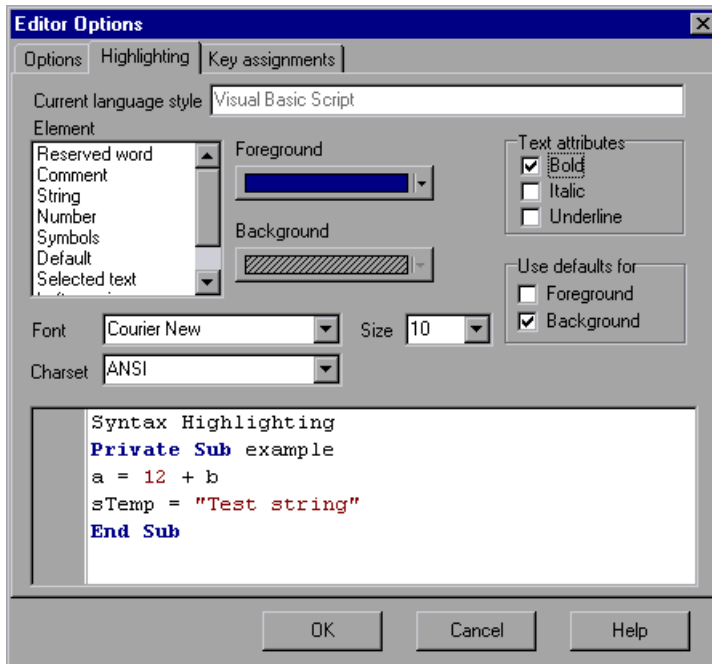
Options	Description
Visible right margin	Displays a line that indicates the Expert View tab's right margin.
Right margin	Sets the position, in characters, of the Expert View tab's right margin (0=left tab edge).
Visible gutter	Displays a blank area (gutter) in the Expert View tab's left margin.
Gutter width	Sets the width, in pixels, of the gutter.
Block indent step size	Sets the number characters that the selected block of VBScript statements will be moved (indented) when the INDENT SELECTED BLOCK softkey is used. For more information on editor softkeys, see "Personalizing Editing Commands".
Tab stop	Sets the distance, in characters, between each tab stop.

Highlighting Script Elements

The Virtual User Recorder tests contain many different elements, such as comments, strings, Astra LoadTest reserved words, operators and numbers. Each element of a Virtual User Recorder test is displayed in a different color and style. You can create your own personalized color scheme and style for each script element. For example, all comments in your scripts could be displayed as italicized, blue letters on a yellow background.

To edit script elements:

- 1 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.

Options	Description
Use defaults for	Applies the font and colors of the “default” style to the selected style.
Font	Sets the font of the script element.
Size	Set the size, in points, of the script element.
Charset	Sets the character subset of the selected font.

An example of each change you apply will be displayed in the pane at the bottom of the dialog box.

- 4 Click **OK** to apply the changes.

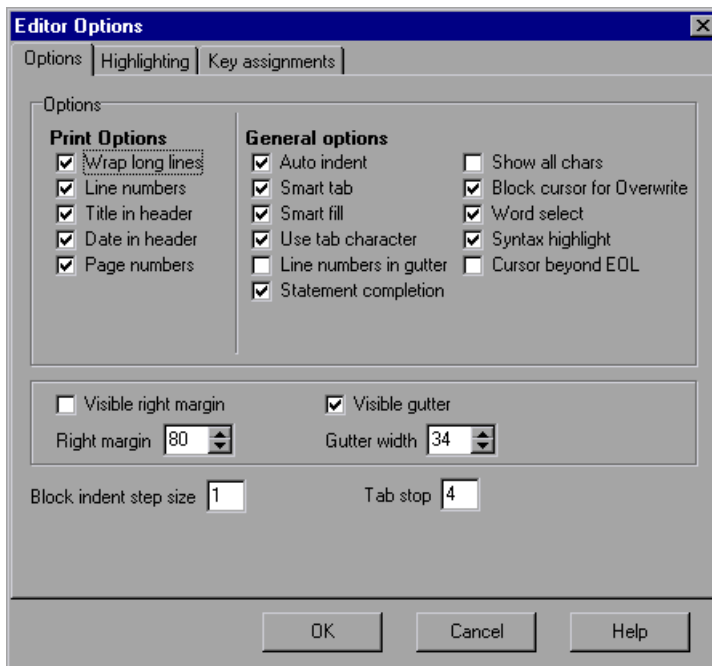
Customizing Print Options

You can set how the hard copy of your script will 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.
Title in header	Inserts the Title into the header of the printed script.
Date in header	Inserts today's date into the header of the printed script.
Page numbers	Numbers each page of the script.

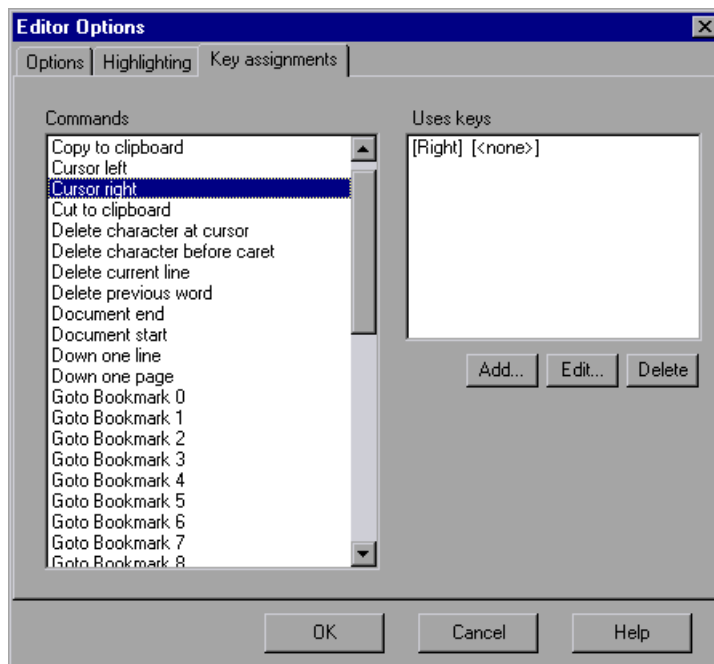
4 Click **OK** to apply the changes.

Personalizing Editing Commands

You can personalize the default keyboard commands you use for editing test scripts. The Virtual User Recorder 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.

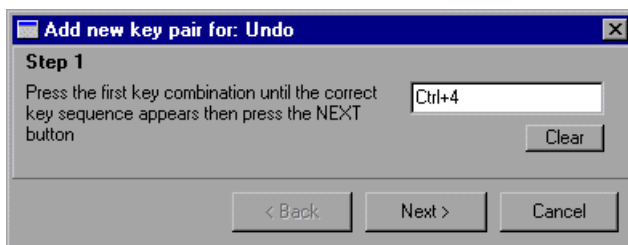
To personalize editing commands:

- 1 In the Expert View, choose **Tools > Editor Options**. The Editor Options dialog box opens.
- 2 Click the **Key Assignments** tab.

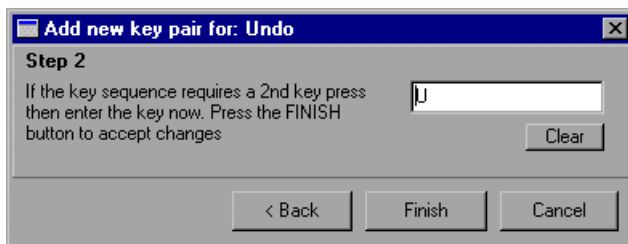


- 3 Select a command from the **Commands** list.

- 4 Click **Add** to create an additional key assignment or click **Edit** to modify the existing assignment. The Add/Edit key pair for dialog box opens. Press the keys you want to use. For example, CTRL + 4.



- 5 Click **Next**. To add an additional key sequence, press the keys you want to use. For example, U.



- 6 Click **Finish** to add the key sequence(s) to the **Use keys** list.

If you want to delete a key sequence from the list, highlight the keys in the **Uses Key** list and click **Delete**.

- 7 Click **OK** to apply the changes.

25

Setting Testing Options from a Test Script

You can control how Astra LoadTest records and runs tests by setting and retrieving testing options from within a test script.

This chapter describes:

- ▶ Setting Testing Options
- ▶ Retrieving Testing Options
- ▶ Controlling the Test Run
- ▶ Adding and Removing Runtime Settings

About Setting Testing Options from a Test Script

Astra LoadTest testing options affect how you record test scripts and run tests. For example, you can set the maximum time that Astra LoadTest allows for finding an object in a page.

You can set and retrieve the values of testing options from within a test script using the **Setting Object** method in the Expert View. For more information on Programming in the Expert View, see Chapter 4, “Testing in the Expert View.”

By retrieving and setting testing options in a test script using the **Setting Object** method, you can control how Astra LoadTest 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 22, “Setting the Virtual User Recorder Testing Options.” For more information on setting options for a single test, see Chapter 3, “Setting Testing Options for a Single Test.”

Setting Testing Options

You can use the **Setting Object** method to set the value of a testing option from within the test script. To set the option, use the following syntax:

```
Setting ( testing_option ) = new_value
```

Some options are global and others are per-test settings.

Using the **Setting Object** with a global testing option changes a testing option globally, and this change is reflected in the Options dialog box.

Note: When using the Setting.Add method, an error occurs if you try to add an existing key value. To avoid this error you should use the Setting.Exist method first. For more details about all the Setting methods, see the Astra LoadTest Function Reference.

For example, if you execute the following statement:

```
Setting("DefaultTimeout")=X
```

The value of *X* replaces the value of the Object Timeout as found in the Properties tab of the Test Settings. The setting remains in effect until it is changed with another **Setting** statement.

Using the **Setting Object** to set per-test options is also 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 344.

For example, if you execute the following statement:

```
Setting("WebTimeOut")=50000
```

Astra LoadTest automatically changes the amount of time it waits for a Web page to load before running a test step to 50000 milliseconds. The setting remains in effect until it is changed again, either with another **Setting** statement, or by setting the **Browser navigation timeout** option in the Web tab of the Test Settings dialog box.

The following section lists some of the Virtual User Recorder testing options that can be used with the Setting object from within a test script. The corresponding dialog box option is listed where applicable.

ObjectTimeOut

Sets or retrieves the delay (in milliseconds) for finding objects.

ObjectTimeOut is a test-specific option.

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 “Properties Testing Options,” on page 310.

NavigationTimeOut

Sets the interval (in milliseconds) Astra LoadTest waits for the Web page to load before running a test step.

NavigationTimeOut 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 311.

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 the Vuser Log and the Test Report, use the syntax:

```
Services.LogMessage ( Setting ( testing_option ) )
```

Controlling the Test Run

You can use the retrieve and set capabilities of the **Setting** object together to control a test run during a current replay. The value you set is in effect from the point where you set the value, until the end of the test replay, or until the specific changed value is reset.

For example, say there is a specific set of steps in your script where you want to increase the Object timeout. You only want those specific steps to be effected and then you want to return to the original timeout value for the duration of the test.

Using the following code, you retrieve the *ObjectTimeOut* value, change the value to 30000 milliseconds, and then return the value to the initial value.

```
Dim initTimeOut  
initTimeOut=Setting("ObjectTimeOut")  
Setting("ObjectTimeOut")=30000  
...  
...  
Setting("ObjectTimeOut")=initTimeOut
```

Adding and Removing Runtime Settings

In addition to the global and test-specific settings, you can also add, modify, and remove your own runtime settings. These settings are applicable during the test run only.

To add a new runtime setting, use the syntax:

```
Setting.Add "testing_option", "value"
```

For example, you could create a setting that indicates the name of the current tester and writes the name in the report.

```
Setting.Add "Tester Name", "Mark Train"  
Services.LogMessage, "Test Run By:", paramcount
```

To modify a runtime setting that has already been initialized, use the same syntax you use for setting any standard setting option:

Setting (*testing_option*) = *new_value*

For example:

```
Setting("Tester Name")="Alice Wonderlin"
```

To remove a runtime setting, use the following syntax:

Setting.Remove (*testing_option*)

For example:

```
Setting.Remove ("Tester Name")
```

Note: You cannot remove the Astra LoadTest predefined run-time settings.

Part VI

Working with TestDirector

26

Working with TestDirector

Web site testing typically involves creating and running many tests. TestDirector, Mercury Interactive's test management tool, can help you organize and control the testing process.

This chapter describes:

- ▶ Using Astra LoadTest with TestDirector
- ▶ Connecting to and Disconnecting from a Project
- ▶ Saving Tests to a Project
- ▶ Opening Tests in a Project
- ▶ Running Tests from TestDirector

About Working with TestDirector

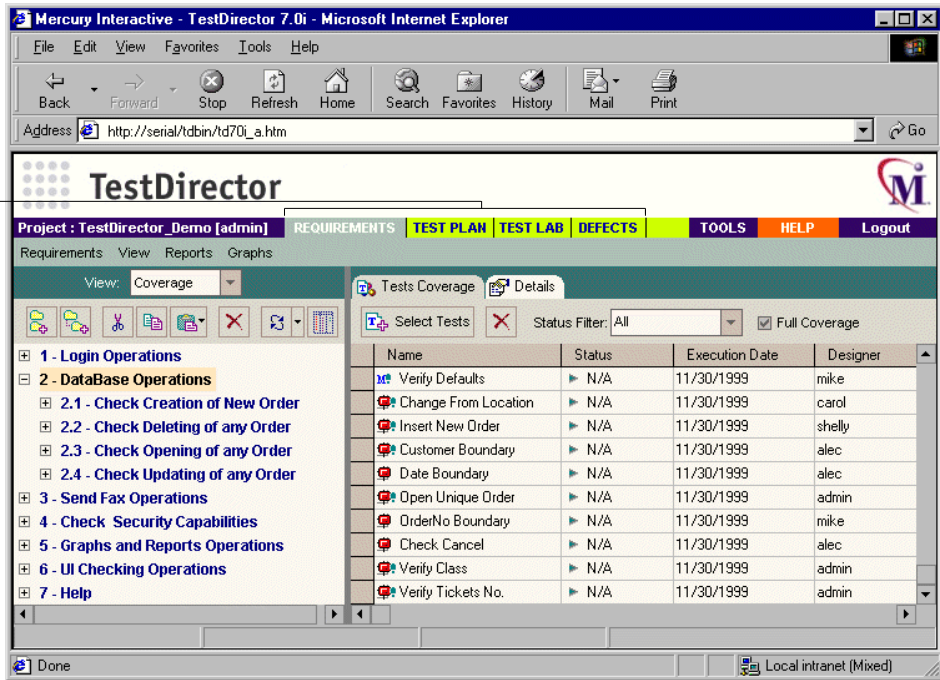
TestDirector is a powerful Web-based test management tool that helps you systematically control the testing process. It helps you create a framework and foundation for your testing workflow.

TestDirector helps you maintain a project of tests that 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



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 LoadTest, and analyzing the plan.

Operation Mode	Description
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 LoadTest, 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*.

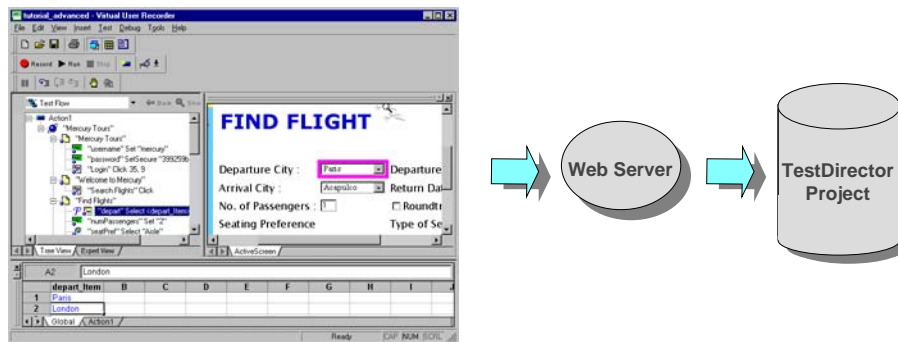
Note: The integration of Astra LoadTest with TestDirector (described in this chapter) is valid only for TestDirector 7.0i.

Using Astra LoadTest 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 LoadTest work together to integrate all aspects of the testing process. In Astra LoadTest, 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 LoadTest to access the project, you must connect it to the Web server where TestDirector is installed.



Astra LoadTest

When Astra LoadTest is connected to TestDirector, you can:

- save a test by associating it with a subject in the Test Plan manager
- schedule to run a test on local or remote hosts. (Test run results are sent directly to your TestDirector project.)
- report defects to a TestDirector project directly from the Test Results window

For information on reporting defects to a TestDirector project directly from the Test Results window, see Chapter 16, “Analyzing Test Results in Stand-Alone Mode.”

Note: The integration of Astra LoadTest with TestDirector, described here, is valid only for TestDirector 6.0 and higher).

Connecting to and Disconnecting from a Project

If you are working with both Astra LoadTest and TestDirector, Astra LoadTest can communicate with your TestDirector project. You can connect or disconnect Astra LoadTest from a TestDirector project at any time during the testing process. However, do not disconnect Astra LoadTest from TestDirector while an Astra LoadTest test is opened from TestDirector.

The connection process has two stages. First, you connect Astra LoadTest to the TestDirector server. This server handles the connections between Astra LoadTest and the TestDirector project.

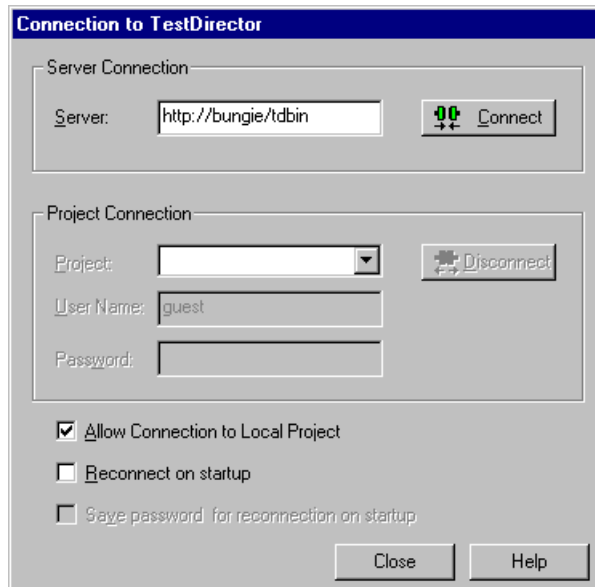
Next, you choose the project you want Astra LoadTest to access. The project stores tests and test run information for the Web site you are testing. Note that TestDirector project databases are password protected, so you must provide a user name and a password.

Connecting Astra LoadTest to TestDirector

You must connect Astra LoadTest to the Web server where TestDirector is installed, before you connect Astra LoadTest to a project. For more information, see “Using Astra LoadTest with TestDirector” on page 352.

To connect Astra LoadTest 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 LoadTest 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 startup** check box is selected, then the **Save Password for Reconnection on startup** check box is enabled. To save your password for reconnection on startup, select the **Save Password for Reconnection on startup** check box. If you do not save your password, you will be prompted to enter it when Astra LoadTest connects to TestDirector on startup.

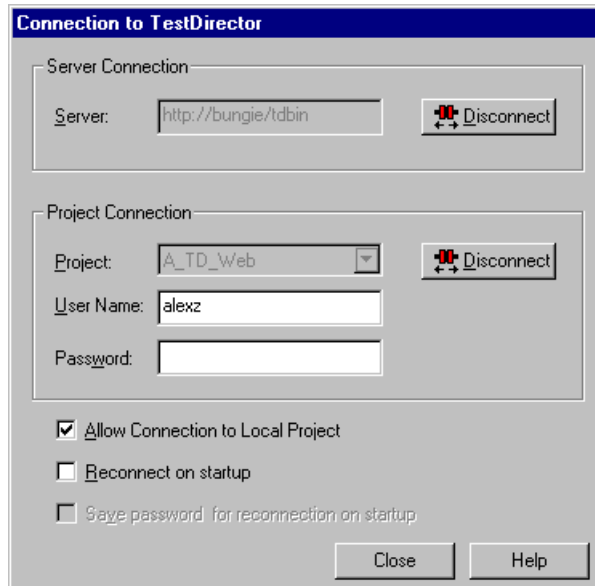
- 8 Click **Close** to close the Connection to TestDirector dialog box.

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 LoadTest 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 LoadTest 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 LoadTest 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 LoadTest from the TestDirector server.
- 3** Click **Close** to close the Connection to TestDirector dialog box.

Note: If you disconnect Astra LoadTest from a TestDirector server without first disconnecting from a project, Astra LoadTest's connection to that database is automatically disconnected.

Saving Tests to a Project

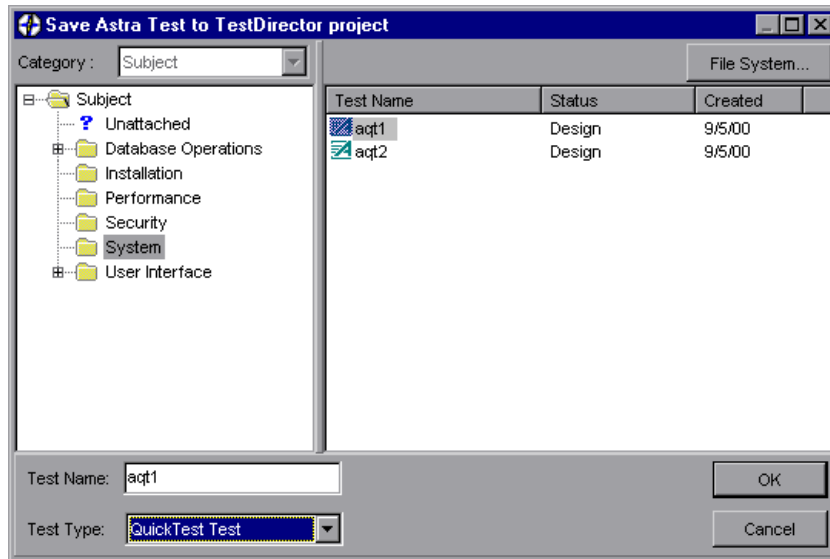
When Astra LoadTest is connected to a TestDirector project, you can create new tests in Astra LoadTest and save them directly to your project. To save a test, you give it a descriptive name and associate it with the relevant subject in the test plan tree. This helps you to keep track of the tests created 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 LoadTest, 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 LoadTest 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 LoadTest 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.

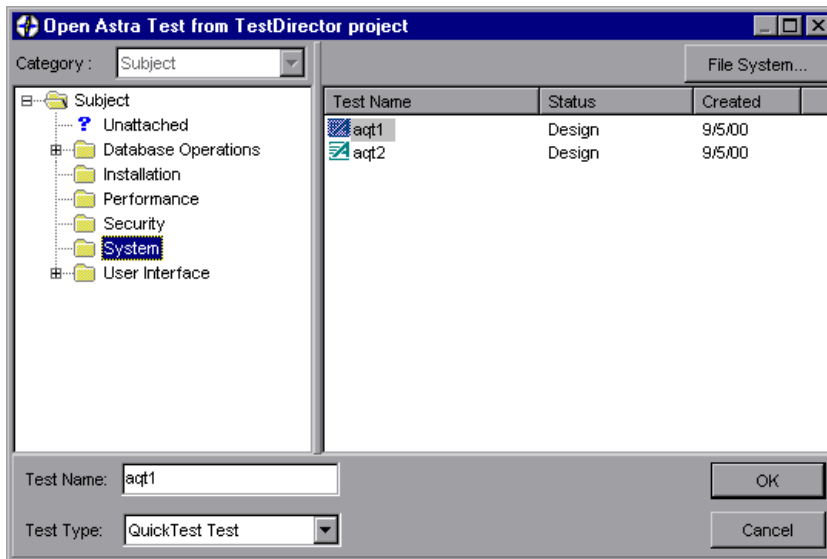
Opening Tests in a Project

If Astra LoadTest 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 LoadTest, 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 LoadTest is connected to a TestDirector project.

To open a test directly from the file system, click the **File System** button to open the Open Astra LoadTest 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.

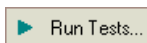
- 3 Select a test in the **Test Name** list. The test appears in the read-only Test Name box.
- 4 Click **OK** to open the test. The test opens in a window in Astra LoadTest.

Running Tests from TestDirector

You can run an Astra LoadTest test from a TestDirector project.

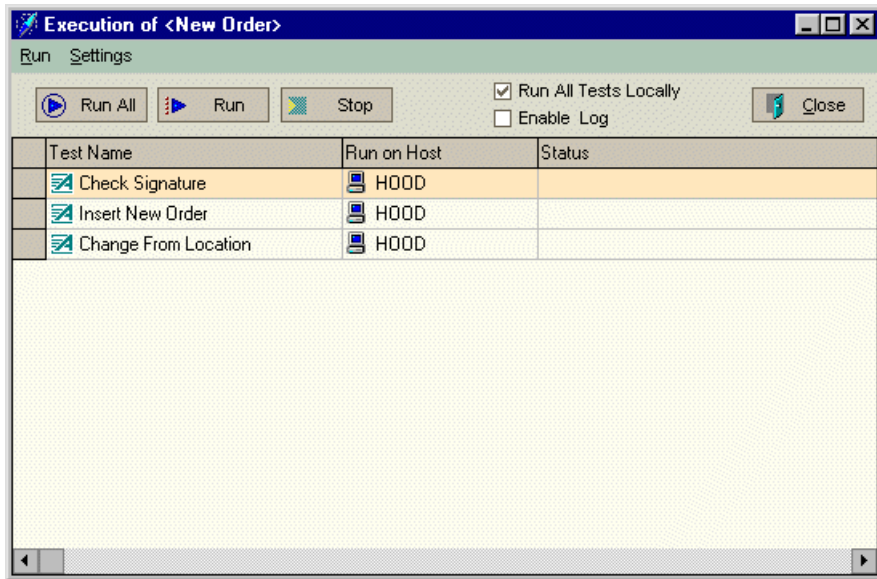
To run a test from a TestDirector project:

- 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**.
- 6 Click **Close** to close the Execution dialog box.

Launch Report

Index

A

- action data sheet 168, 191
- Action List 169
- Action tab, Data pane 13, 168
- Action toolbar 169
- action tree, defined 169
- actions 165–187
 - creating new 170
 - diagram 166, 172
 - existing, inserting 172
 - guidelines for working with 186
 - inserting a call to 175
 - inserting a copy of 173
 - multiple actions in a test 166–167
 - nesting 177
 - non-reusable, defined 167
 - overview 166
 - parameterizing 166–167
 - removing from a test 184
 - reusable, defined 167
- Activate Pointed Window After box 308
- Active Server Page technology 298
- ActiveScreen
 - changing 34
- ActiveScreen tab
 - in Display pane 12
 - in Options dialog box 309
- ActiveX
 - inserting functions 100
- ActiveX Wizard 100
- Add Defect button, in Test Results window 234
- Add/Remove Properties dialog box 43, 85, 151
- adding checkpoints
 - objects 70–74
 - pages 55–65
 - tables 75, 75–78
 - text 65–69
- advanced issues
 - dynamic Web content 296
 - maintaining tests 299
 - running tests 295–296
 - testing localized applications 300
 - Web issues 298–299
- Alternative Navigation Properties dialog box 322
- analyzing test results 225–234
 - checkpoints 230
 - filtering results 229
 - printing results 233
 - Runtime Data table 232
 - Test Results button 228
 - Test Results window 226
- application, sample x, 8
- applications, testing localized versions 300
- ASCII 192
- asp files 298
- Astra LoadTest
 - Data pane 10
 - Debug toolbar 10, 15
 - Display pane 10
 - File toolbar 9, 14
 - introduction 3–8
 - Load toolbar 10, 15
 - Main toolbar 10, 14
 - menu bar 9
 - overview 9–18
 - resources x
 - starting 9
 - status bar 10
 - Test pane 10
 - title bar 9
 - window 9

- Astra LoadTest context sensitive help x
- Astra LoadTest Readme file x
- Astra LoadTest Tutorial x
- Astra LoadTest Virtual User Recorder User's Guide 3
- Astra LoadTest window, Display pane 12
- Astra QuickTest help x
- Astra QuickTest Readme file x
- Astra QuickTest Tutorial x
- authentication 320
- AutoFill List command, data table 195
- automatic think time 119

B

- Basic event recording configuration level 246
- behavior, definition 253
- books online x
- breakpoints
 - Clear All Breakpoints button 238
 - deleting 238
 - Insert/Remove Breakpoint button 238
 - overview 235
 - setting 237
 - Toggle Breakpoint button 238
- Browser Configuration tab, Run-Time
 - Settings dialog box 122
- browsers, supported 53
- bubbling, definition 254
- buttons, Astra LoadTest toolbars 13

C

- calculations, in the Expert View 284
- CGI scripts 298
- checking Web objects 53–78
- Checkpoint command 17
- Checkpoint Properties dialog box 49, 72
- checkpoints 53–78
 - checking objects 70–74
 - checking pages 55–65
 - checking tables 75–78
 - checking text 65–69
 - Checkpoint Properties dialog box 72
 - definition 26, 47
 - in the Expert View 274

- modifying 51
- Page Checkpoint Properties dialog box 55, 56
- parameterizing 89–90
- Table Checkpoint Properties dialog box 76
- Text Checkpoint Properties dialog box 66
- Use Data Table Formula option 201
- Clear All Breakpoints button 15, 238
- Clear All Breakpoints command 18
- Clear command, data table 194
- Collapse All command 229
- COM 299
- Command tab
 - Debugger view 240
- commands using shortcut keys 16–18
- comments
 - in the Expert View 283
 - in the Tree View 269
- Complete Word command 16
- conditional statements 264
- configuration levels
 - custom 248–254
 - standard 246–247
- configuring event recording 245–256
- connecting Astra LoadTest to a TestDirector project 354–358
- Connection to TestDirector dialog box 355
- conventions. *See* typographical conventions
- cookies 298
- Copy command 16
 - data table 193
- creating a new test 35
- creating checkpoints 47–51
- creating tests 21–36
- Custom event recording configuration
 - adding listening events 251
 - adding objects to the custom list 250
 - deleting objects from the custom list 251
 - setting 248
 - specifying listening criteria 253
- custom event recording configuration 248–254

- custom web event configuration files
 - loading 255
 - saving 255
- Custom Web Event Recording Configuration
 - dialog box 249
- customizing test scripts 331–340
 - highlighting script elements 335
 - overview 331
 - print options 337
 - script window customization 338
- Cut command 16
 - data table 193

D

- Data menu commands, Data table 194
- Data pane 10, 13
 - Action tab 13, 168
 - Global tab 13, 167
 - table columns 80
 - table rows 80
- data sheets
 - global 190
 - global and action, choosing between 167
 - local 191
- data table 189–204
 - AutoFill List command 195
 - Clear command 194
 - Copy command 193
 - Currency(0) command 195
 - Currency(2) command 195
 - Custom Number command 196
 - Cut command 193
 - Data menu commands 194
 - data sheets 190
 - Date (M/d/yy) command 195
 - Delete command 194
 - Edit menu commands 193
 - editing tables 191–196
 - Export command 193
 - File menu commands 193
 - Fill Down command 194
 - Fill Right command 194
 - Find command 194
 - Fixed command 195

- Format menu commands 195
- Fraction command 195
- General command 195
- Go To command 194
- Import command 193
- Import from command 195
- importing data in ASCII 192
- importing data in Microsoft Excel 95
 - 192
- importing data in Microsoft Excel 97
 - 192
- Insert command 194
- local data sheets 191
- location 191
- menu commands to edit tables 193
- Paste command 193
- Paste Values command 194
- Percent command 195
- Print command 193
- Recalc command 194
- Replace command 194
- Scientific command 195
- scripting functions, using 204
- Sort command 194
- Time (hmm AM/PM) command 195
- using formulas in 200–202
- Validation Rule command 196
- worksheet functions 201
- See also* Data pane
- Data Views button 14
- databases
 - creating a query in ODBC/Microsoft Query 199–200
- debug mode 216
- Debug toolbar, Astra LoadTest window 10, 15
- Debugger view 239–240
 - Command tab 240
 - Variables tab 240
 - Watch Expressions tab 240
- debugging tests 235–242
 - deleting breakpoints 238
 - example 241
 - overview 235
 - pausing runs 237
 - setting breakpoints 237

debugging tests - continued

- Step Into button 236
- Step Out button 236
- Step Over button 237
- default optional steps 222
- default properties, modifying 37–46
- defects, reporting from test results 233
- Delete command 16
 - data table 194
- deleting actions 184
- deleting breakpoints 238
- descriptions, modifying 41–44
- descriptive programming 279–283
 - syntax 280
 - using the Index property in 282
 - using the With statement in 281
 - using variables with 280
 - WebElement object 281
- Dim statement, in the Expert View 288
- disconnecting from a TestDirector
 - project 357
 - server 358
- Display pane 10, 12
 - ActiveScreen tab 12
- Display Views button 14
- Do...Loop statement, in the Expert View 286
- Document Object Model 291
- DOM 291
- DOM additional information 293
- dynamic descriptions of objects 41
- dynamic parameters 296
- dynamic Web content 296
- dynamically generated
 - URLs 296
 - Web pages 296

E

- Edit menu commands, data table 193
- encoding passwords 203
- event configuration 245–256
- Excel formulas 201
- exception configuration file 211, 300
- Exception Editor 205
- exception handling 205–212
 - adding new exceptions 210

- changing status of exceptions 207
- configuring 211, 300
- deleting exceptions 211
- Exception Editor 205
 - modifying exceptions 208
- Exception.inf file 211, 212, 300
- Exist function 296
- existing actions, inserting 172
- Expand All command 229
- expected value, modifying 50
- Expert View 271–293
 - checkpoints 274
 - parameters 274
 - viewing steps 272
- Expert View tab 11
- Expert View tab, Test pane 11
- Export
 - test to a zip file 16
- Export command, data table 193
- eXtensible Markup Language 299

F

- FAQs 295–301
- File menu commands, data table 193
- File toolbar, Astra LoadTest window 9, 14
- Fill Down command, data table 194
- Fill Right command, data table 194
- Filter button, in Test Results window 229
- Filter Image Check dialog box 63
- Filter Link Checks dialog box 61
- filtering
 - hypertext links to check 61
 - image sources to check 63
- Find command 16
 - data table 194
- For...Each statement, in the Expert View 286
- For...Next statement, in the Expert View 285
- Format menu commands, data table 195
- formulas
 - using in checkpoints 201
 - using in the data table 200–202
 - using to create input parameters 201
- frequently asked questions 295–301
- Function Arguments command 17
- Function Arguments dialog box 87

Function Reference, Astra x
 Function wizard 257, 258–263
 functions
 in the Tree View 258–263

G

General tab, Options dialog box 307
 GetSavedResponseData 297
 global and action data sheets, choosing
 between 167
 global data sheet 167, 190
 global parameter, defined 83
 Global tab, Data pane 13, 167
 Go To command 16
 data table 194
 GUI Spy 38

H

handler, definition 253
 handling exceptions 205–212
 adding new exceptions 210
 changing status of exceptions 207
 deleting exceptions 211
 Exception Editor 205
 modifying exceptions 208
 help x
 High event recording configuration level 247
 HTTP requests 328
 hypertext links, filtering 61

I

identifying objects 37–46
 If Statement dialog box 266
 If...Then...Else statement, in the Expert View
 287
 image sources, filtering 63
 Import
 test from a zip file 16
 Import command, data table 193
 Import from command, data table 195
 Index property, descriptive programming
 and 282
 Insert Action dialog box 176
 Insert Checkpoint button 14

Insert command, data table 194
 Insert Copy of Action dialog box 174
 Insert New Action dialog box 171
 Insert/Remove Breakpoint button 238
 Insert/Remove Breakpoint command 18
 inserting
 transactions 110
 inserting functions
 ActiveX 100
 inserting, Rendezvous points 112
 iteration
 defined 189
 Iterations tab, Test Settings dialog box 315

J

Java applets
 testing 107

K

key assignments, creating 339
 keyboard shortcuts
 creating 339
 deleting 339
 editing 339

L

Load ActiveX Controls check box 310
 Load Images check box 309
 Load Java Applets check box 309
 load mode 216
 load testing 109
 rendezvous points 112
 transactions 109
 Load toolbar, Astra LoadTest window 10, 15
 Load vuser log viewer 218
 local data sheet
 See action data sheet
 local data sheets 191
 local parameter, defined 83
 localization 191
 localized applications, testing 300
 Log options, Run-Time Settings dialog box
 116

M

- Main toolbar, Astra LoadTest window 10, 14
- managing tests 35–36
 - advanced issues 299
 - creating new 35
 - opening 35
 - printing 36
 - saving 35
 - unzipping 36
 - zipping 36
- managing the testing process 8, 349–362
- mathematical formulas 201
- Medium event recording configuration level 247
- menu bar, Astra LoadTest window 9
- Mercury Interactive on the Web xi
- Mercury Tours sample application x, 8, 153
- meta tags 298
- methods
 - of runtime objects 290
 - viewing 37–46
- Microsoft Excel 192, 200
- Microsoft Internet Explorer 54
- Microsoft Query
 - choosing a database for a database checkpoint 199–200
- Microsoft Visual Basic Scripting language 7
- modifying
 - default properties 37–46
 - descriptions 41–44
 - steps 28–29
- multiple actions in a test 166–167

N

- nesting actions 177
- Netscape Navigator 53
- Network tab, Run-Time Settings dialog box 120
- New Action button 14, 171
- New Action command 171
- New button 14
- New command 16
- non-reusable action, defined 167
- NTLM authentication 320

O

- object descriptions, modifying 41–44
- object methods
 - runtime 290
- object properties
 - runtime 290
- Object Properties command 17
- Object Properties dialog box 29, 43, 84, 150
- object properties, viewing with the GUI Spy 38
- Object property 291
- Object Repository dialog box 29
- Object Repository, deleting an object from 33
- objects
 - checking 70–75
 - dynamic descriptions 41
 - identifying 37–46
 - viewing methods 37–46
- ODBC
 - choosing a database for a database checkpoint 199–200
- On Run Error box 308
- Open Astra Test from TestDirector Project dialog box 360
- Open button 14, 24, 35
- Open command 16
- Open dialog box 35
- opening tests 35
 - in a TestDirector project 360
- optional steps 221–223
 - default 222
 - setting 221
- options
 - setting testing options for a single test 311–330
- Options dialog box
 - ActiveScreen tab 309
 - General tab 307
- options, testing
 - See setting testing options
- Output messages 126
- Output Parameter command 17
- Output Parameter Properties dialog box 143

- output parameters 129–147
 - definition 129
 - objects 141–144
 - Output Parameter Properties dialog box 143
 - Page Output Parameter Properties dialog box 133
 - pages 131–135
 - Table Output Parameter Properties dialog box 146
 - tables 145–147
 - text 135–140
 - Text Output Parameter Properties dialog box 137

P

- Page Checkpoint Properties dialog box 56, 57
- Page Output Parameter Properties dialog box 133
- Page properties
 - alternative navigation properties 322
- Parameter Info command 16
- parameterization 79–96
 - checkpoints 89–90
 - example 91–96
 - overview 79
 - steps 83–88
- parameterization (output) 129–147
 - objects 141–144
 - Output Parameter Properties dialog box 143
 - Page Output Parameter Properties dialog box 133
 - pages 131–135
 - Table Output Parameter Properties dialog box 146
 - tables 145–147
 - text 135–140
 - Text Output Parameter Properties dialog box 137
- parameterized test, example 91–96
- parameterizing
 - actions 166–167
- parameters, Expert View 274

- passing data between actions 167
- Password Encoder dialog box 203
- password encoding 203
- Paste command 16
 - data table 193
- Paste Values command, data table 194
- Pause button 15, 221, 237
- Pause command 17
- pausing test runs 237
- planning tests 22–23
- power users 295
- Print button 14, 36
- Print button, in Test Results window 233
- Print command 16
 - data table 193
- print options 337, 338
- printing tests 36
- programming
 - adding functions to tests 258–263
 - comments 269
 - conditional statements 264
 - Expert View 271–293
 - Function wizard 257, 258–263
 - in the Expert View 284
 - sending messages to test results 268
 - Tree View 257–269
- project (TestDirector)
 - disconnecting from a 357
 - opening tests in a 360
 - running tests from a 361
 - saving tests to a 358–359
- properties
 - of runtime objects 290
- Properties tab, Test Settings dialog box 318
- properties, default 37–46

Q

- query file for a database checkpoint, working with ODBC/Microsoft Query 199–200

R

- Readme file x
- Recalc command, data table 194
- Record button 14, 24

- Record command 17
- recording
 - on Web sites 53
- Recording status options 254
- recording tests 23–25
- redirection of server 298
- regular expressions 149–164
 - defining in object checkpoints 153
 - defining in steps 150
 - defining in text checkpoints 156
 - overview 149
 - syntax 159
 - treating special characters literally
 - 152, 155, 158
- removing actions 184
- Rename Action command 186
- Rename command 17
- rendezvous points 112
- Replace command 16
 - data table 194
- Replace with Alternative Navigation
 - Properties 325
 - parameterizing 326
- reporting defects from the Test Results window 233
- resetting standard event recording
 - configuration settings 255
- resources
 - Astra Function Reference x
 - Astra LoadTest context sensitive help
 - x
 - Astra LoadTest Readme file x
 - Astra LoadTest Tutorial x
 - Astra QuickTest help x
 - Astra QuickTest Readme file x
 - Astra QuickTest Tutorial x
 - books online x
 - Mercury Interactive on the Web xi
 - support information xi
 - technical support online xi
 - VBScript reference guide xi
- reusable action, defined 167
- Run button 14, 217, 220, 237
- Run command 17
- Run Mode box 308
- Run Scripts check box 310
- Run Test dialog box 217, 220
- running a test remotely
 - from TestDirector 308
- running single actions 167
- running tests 215–223
 - advanced issues 295–296
 - from a TestDirector project 361
 - on Web sites 53
 - overview 215
 - Pause button 221
 - Run button 217, 220
 - Run Test dialog box 217, 220
 - running an entire test 216–221
 - Stop button 221
 - using optional steps 221–223
 - viewing test results 227
- running transaction files
 - Run Test dialog box 217, 220
- Runtime Data table, Test Results window 227
- runtime object
 - methods 290
 - properties 290
- run-time settings 114
 - Advanced options 117
 - Browser Configuration tab 122
 - Error handling options 117
 - Log options 116
 - Network tab 120
 - Think Time tab 119
- Run-time Settings dialog box 114
- run-time settings, adding and removing 344

S

- sample application, Mercury Tours x, 8, 153
- Save Astra Test to TestDirector Project dialog box 359
- Save button 14, 35
- Save command 16
- Save dialog box 35
- SaveResponseData 297
- saving tests 35
- saving tests to a TestDirector project
 - 358–359
- sending messages, output 126

- server (TestDirector), disconnecting from a 358
 - server redirections 298
 - server side connections 298
 - session IDs 298
 - setting
 - ActiveScreen options in the Options dialog box 309
 - breakpoints 237
 - Browser Configuration options in the Run-Time Settings dialog box 122
 - Browser options in the Test Settings dialog box 314
 - General options in the Options dialog box 307
 - General options in the Test Settings dialog box 115
 - Iterations options in the Test Settings dialog box 315
 - Log options in the Run-Time Settings dialog box 116
 - Network options in the Run-Time Settings dialog box 120
 - Properties options in the Test Settings dialog box 318
 - run-time options 114
 - test preferences options 311–330
 - testing options 311–313
 - Think Time options in the Run-Time Settings dialog box 119
 - User Information options in the Test Settings dialog box 320
 - Web options in the Test Settings dialog box 319
 - Setting object 342
 - setting optional steps 221
 - setting testing options
 - See also* testing options
 - setting testing options, within a test script 341–345
 - SGML 299
 - shortcut keys 16–18
 - Sort command, data table 194
 - spawned windows 296
 - standard event recording configuration 246–247
 - Start Recording dialog box 24
 - StartUp tab, Test Settings dialog box 314
 - status bar, Astra LoadTest window 10
 - Step commands 236
 - Step Into button 15, 236
 - Step Into command 17
 - Step Out button 15, 236
 - Step Out command 18
 - Step Over button 15, 237
 - Step Over command 17
 - steps
 - modifying 28–29
 - steps, optional 221–223
 - Stop button 14, 25, 221
 - Stop command 17
 - support information xi
 - supported Web browsers 53
 - Sync on Request 328
 - synchronization 328
- T**
- Table Checkpoint Properties dialog box 76
 - Table Output Parameter Properties dialog box 146
 - technical support online xi
 - test
 - checking tables 75
 - managing 35–36
 - opening 35
 - parameterization 79–96
 - parameterization (output) 129–147
 - planning 22–23
 - printing 36
 - programming 257–269
 - recording 23–25
 - saving 35
 - unzipping 36
 - zipping 36
 - test flow, defined 169
 - Test Object Repository, finding an object property or value 30
 - Test Object Repository, replacing an object property value 32

- Test pane 10, 11
 - Expert View tab 11
 - Tree View tab 11
- test replay
 - tuning 322
- test results
 - reporting defects 233
- Test Results button 228
- Test Results window 226
 - Runtime Data table 227
 - test results details 227
 - test results tree 226
- test results, sending messages to 268
- test scripts
 - customizing 331–340
 - highlighting script elements 335
 - print options 337
 - script window customization 338
- Test Settings dialog box
 - Browser tab 314
 - Iterations tab 315
 - Properties tab 318
 - User Information tab 320
 - Web tab 319
- test synchronization 328
- test tree
 - creating 24
 - definition 27
- test window
 - customizing appearance of 332
 - highlighting script elements 335
- TestDirector
 - managing the testing process 8
 - modes of operation 350
 - running a remote test 308
 - using Astra LoadTest with 8
 - working with 349–362
- TestDirector Connection button, in Test Results window 233
- TestDirector project
 - connecting Astra LoadTest to a 354–358
 - disconnecting from a 357
 - opening tests in a 360
 - running tests from a 361
 - saving tests to a 358–359
- TestDirector server
 - connecting Astra LoadTest 354
 - disconnecting from a 358
- testing
 - Java Applets 107
- testing in Expert View 271–293
- testing load 109–127
- testing options
 - restoring 343
 - retrieving 343
 - run-time 344
 - setting 342
 - setting for a single test 311–330
 - within a test script 341–345
- testing process 4
 - analyzing test results 7
 - creating tests 4
 - running tests 6
- tests
 - checking objects 70–74
 - checking pages 55–65
 - checking Web objects 53–78
 - debugging 235–242
 - diagram 166, 172
 - multiple actions in 166–167
 - pausing runs 221, 237
 - printing results 233
 - running 215–223
 - test results 225–234
- Text Checkpoint Properties dialog box 66, 157
- TEXT function in data table worksheet 200
- Text Output Parameter Properties dialog box 137
- Think Time tab, Run-Time Settings dialog box 119
- title bar, Astra LoadTest window 9
- Toggle Breakpoint button 15, 238
- toolbars, Astra LoadTest window
 - Debug 10, 15
 - File 9, 14
 - Load 10, 15
 - Main 10, 14
- transactions 109
 - inserting 110
- Tree View tab, Test pane 11

- tuning
 - test replay 322
- Turboload 123
- tutorial x
- typographical conventions in this guide xii

U

- unzipping tests 36
- Use Data Table Formula option 201
- User Information tab, Test Settings dialog box 320

V

- VALUE function in data table worksheet 200
- Variables tab
 - Debugger view 240
- VBScript
 - Astra Functions x
 - reference guide xi
- View Results when Test Run Ends check box 308
- viewing test results 225–234
 - checkpoints 230
 - filtering results 229
 - printing test results 233
 - Runtime Data table 232
 - Test Results button 228
 - Test Results window 226

W

- Watch Expressions tab
 - Debugger view 240
- Web browsers, supported 53
- Web content accessibility checkpoints 65
 - creating automatic 65
 - creating individual 65
 - setting preferences 65
- Web content, dynamic 296
- Web event configuration
 - custom 248–254
 - standard 246–247
- Web event recording configuration 245–256
- Web Event Recording Configuration dialog box 247

- Web sites, recording and running tests 53
- Web tab, Test Settings dialog box 319
- Web, advanced issues 298–299
- WebElement object
 - descriptive programming 281
- While statement, in the Expert View 287
- windows, spawned 296
- WinInet 123
- With statement, in the Expert View 288
- worksheet functions in the data table 201

X

- XML 299

Z

- zipping tests 36



MERCURY INTERACTIVE

Mercury Interactive Corporation

1325 Borregas Avenue
Sunnyvale, CA 94089 USA

Main Telephone: (408) 822-5200

Sales & Information: (800) TEST-911

Customer Support: (877) TEST-HLP

Fax: (408) 822-5300

Home Page: www.mercuryinteractive.com

Customer Support: support.mercuryinteractive.com



* AL TUG5. 4. 3/ 01 *